Modellierung und Erkennung von Kontextinformationen und Nutzerintentionen in Fahrzeuginformationssystemen

Von der
Carl-Friedrich-Gauß-Fakultät
der Technischen Universität Carolo-Wilhelmina zu Braunschweig

zur Erlangung des Grades eines
Doktoringenieurs (Dr.-Ing.)

genehmigte Dissertation

von Daniel Lüddecke
geboren am 11. Juli 1988
in Halberstadt

Eingereicht am:	21.10.2016
Disputation am:	09.12.2016
1. Referentin/Referent:	Prof. Dr.-Ing. Ina Schaefer
2. Referentin/Referent:	Associate Professor Dr. rer. nat. Christian Berger

2016

Technische Universität Carolo-Wilhelmina zu Braunschweig

Carl-Friedrich-Gauß-Fakultät

Dissertation

Modellierung und Erkennung von Kontextinformationen und Nutzerintentionen in Fahrzeuginformationssystemen

Autor:

Daniel Lüddecke

Oktober 2016

Betreuer:

Prof. Dr.-Ing. Ina Schaefer

Institut für Softwaretechnik und Fahrzeuginformatik

Technische Universität Braunschweig

Associate Professor Dr. rer. nat. Christian Berger

Department of Computer Science and Engineering

University of Gothenburg, Sweden

Lüddecke, Daniel:
Modellierung und Erkennung von Kontextinformationen und Nutzerintentionen in Fahrzeuginformationssystemen
Dissertation, Technische Universität Carolo-Wilhelmina zu Braunschweig, 2016.

Inhaltsangabe

Aktuelle Fahrzeuginformationssysteme bieten ihren Nutzern eine Vielfalt an Funktionen, wie die Auswahl eines gewünschten Radiosenders, das Abspielen einer Musikplaylist oder das Starten einer Navigation. Aktuell steht Nutzern dabei zu jeder Zeit der volle Funktionsumfang des Systems zur Verfügung.

Um die wachsende Anzahl an Funktionen für den Nutzer bedienbar zu halten, müssen zukünftige Fahrzeuginformationssysteme in der Lage sein, die vom Nutzer gewünschte Funktion zu erkennen und den Nutzer dabei zu unterstützen diese Funktion schnell zu erreichen. Die vom Nutzer gewünschte Funktion kann dabei von der aktuellen Situation in der sich der Nutzer befindet abhängen, weshalb das Fahrzeuginformationssystem in der Lage sein muss die aktuelle Situation zu erkennen und sein Systemverhalten entsprechend zu adaptieren.

Im ersten Teil der vorliegenden Arbeit wird gezeigt, wie Fahrzeuginformationssysteme im Rahmen eines Modell-basierten Entwicklungsprozesses und unter Zuhilfenahme von Ontologien, basierend auf einer Vielzahl an Kontextinformationen ein Situationsverständnis erhalten können.

Der zweite Teil der Arbeit zeigt anschließend auf, wie Bayes'sche Netze genutzt werden können um individuelle Bedürfnisse und Gewohnheiten der Nutzer dabei zu berücksichtigen.

Die vorgestellten Ansätze und Technologien werden auf die zu Beginn erhobenen Anforderungen hin evaluiert. Die implementierten Verfahren werden hinsichtlich ihrer Leistung untersucht, mit dem Ziel die Leistungsfähigkeit der Modelle zu maximieren und so für ein bestmögliches Laufzeitverhalten zu sorgen.

Vorwort

Diese Dissertation entstand während meiner Tätigkeit als Doktorand in der Konzernforschung der Volkswagen AG. Danken möchte an dieser Stelle denjenigen Personen, die an der Entstehung der in dieser Arbeit präsentierten Inhalte beteiligt waren.

Mein erster Dank geht an meine Betreuerin Prof. Dr.-Ing. Ina Schaefer, die mich von der Vorbereitung, über die Durchführung, bis hin zur Finalisierung der Arbeit begleitete und konstruktiv unterstütze. Darüber hinaus möchte ich Dr.-Ing. Christoph Seidl für die überaus kompetente Mitarbeit an den im Rahmen dieser Arbeit veröffentlichten Konferenz- und Zeitschriftenartikel danken. Mein Dank geht auch an Associate Professor Dr. rer. nat. Christian Berger für die Verfassung des Zweitgutachtens, sowie an Prof. Dr.-Ing. Lars Wolf für die Übernahme des Vorsitzes der Prüfungskommission.

Ebenfalls möchte ich mich bei Dr.-Ing. Marius Spika bedanken, der mir als Betreuer seitens Volkswagen mit seiner wissenschaftlichen Erfahrung und Kompetenz zur Seite stand und damit maßgeblich zur Entstehung dieser Arbeit beigetragen hat. Auch bei meinem Freund und Kollegen Jens Schneider möchte ich mich an dieser Stelle bedanken. In zahlreichen fachlichen Diskussionen konnte ich Fragestellungen, Konzepte und Bewertungen kritisch mit ihm eruieren.

Außerdem möchte ich mich bei allen Studenten bedanken, deren Bachelor-, Master-, und Studienarbeiten ich betreuen durfte und die damit einen ebenfalls nicht unerheblichen Beitrag zu dieser Arbeit geleistet haben. Allen vorweg danke ich Nina Bergmann, die ausgezeichnete Vorarbeiten zur Modellierung von Kontextinformationen leistete. Auch Andreas Pätzold und Stephan Reinert leisteten beeindruckende Arbeit auf den Gebieten der Tool-Unterstützung bzw. der Modellierung von Nutzerintentionen.

Mein wohl größter Dank gebührt meiner Freundin Maria Müller, die mich durch ein jederzeit ausgeglichenes privates Umfeld dabei unterstütze, das langfristige Ziel nicht aus den Augen zu verlieren.

Erklärung

Die Ergebnisse, Meinungen und Schlüsse dieser Arbeit sind nicht notwendigerweise die der Volkswagen AG.

Declaration

The results, opinions and conclusions expressed in this thesis are not necessarily those of Volkswagen AG.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Quelltextverzeichnis

Abkürzungsverzeichnis

BMA	Bayesian Model Averaging
CAN	Controller Area Network
CC/PP	Composite Capabilities/Preference Profiles
CML	Context Modeling Language
DTD	Document Type Definition
EXLAP	eXtensible Lightweight Asynchronous Protocol
FIS	Fahrzeuginformationssystem
HMI	Human-Machine Interface
IEEE	Institute of Electrical and Electronics Engineers
INIS	Intention-aware In-Car Infotainment System
KNN	Künstliches Neuronales Netz
LIN	Local Interconnect Network
LKW	Lastkraftwagen
MOST	Media Oriented Systems Transport
ORM	Object-Role Modelings
OSGi	Open Services Gateway initiative
PKW	Personenkraftwagen
POI	Point of Interest
RDF	Resource Description Framework
SGML	Standard Generalized Markup Language

SWRL	Semanic Web Rule Language
UML	Unified Modeling Language
XML	Extensible Markup Language

1. Einführung

„Try to make sense of what you see and wonder about what makes the universe exist. Be curious, and however difficult life may seem, there is always something you can do, and succeed at.
It matters that you don't just give up.“

(Stephen Hawking, britischer Astrophysiker, [Haw12])

Fahrzeuginformationssystem (FIS) haben sich in den letzten Jahren teils dramatisch im Hinblick auf ihren Funktionsumfang verändert. Während sie in den 1980er Jahren im wesentlichen zum Radio hören, oder zum Abspielen einer CD oder Kassette verwendet wurden, unterstützen sie ihren Nutzer heutzutage durch deutlich komplexere Subsysteme, wie beispielsweise einer staufreien Navigation, der Anbindung einer W-LAN-fähigen Kamera, oder dem Abspielen von Musik aus einer Vielzahl an Streaming-Diensten. Dabei ist davon auszugehen, dass die Anzahl an Features, die Nutzer von ihrem FIS erwarten, stetig steigen wird[1]. Daher ist es unabdingbar, Bedienkonzepte, Systemarchitekturen und Software-Entwicklungsprozesse zu erforschen, die es erlauben, funktionsreiche FIS zu entwickeln und zu bedienen.

In der Konzernforschung der Volkswagen AG werden, mit bewusster Distanz zur Serienentwicklung, Ideen und Konzepte für zukünftige Fahrzeuge des Volkswagen Konzerns entwickelt und untersucht. Im Bereich der FIS zählen dazu neue Bedienkonzepte, die es erlauben, das System verständlich und möglichst intuitiv bedienen zu können. Dazu werden, vor einem möglichen Serieneinsatz, prototypische Umsetzungen solcher Systeme, zum Beispiel durch Nutzerstudien, oder Expertenbefragungen, evaluiert. Um eine solche Evaluation durchführen zu können, müssen neue Bedienkonzepte möglichst schnell und ohne allzu großen Aufwand umgesetzt und in ein Fahrzeug integriert werden können.

[1] http://www.consumerreports.org/cro/magazine/2013/04/connect-with-your-car. Zuletzt überprüft am 19.10.2016.

Diese neuen Bedienkonzepte gehen davon aus, dass funktionsreiche FIS genau dann gut bedienbar bleiben, wenn sie in die Lage versetzt werden, die Intentionen ihrer Nutzers zu erkennen. Dabei genügt es jedoch nicht, dieses Erkennung a posteriori, also nach der Bedienung durch den Nutzer durchzuführen. Vielmehr muss das System zur Laufzeit, also während der Nutzer es bedient, verstehen, welche Ergebnisse der Nutzer zu erreichen versucht. Analog zur zwischenmenschlichen Kommunikation gelingt diese Erkennung genau dann am besten, wenn das System in der Lage ist, die aktuelle Situation, in der sich der Nutzer und das Fahrzeug aktuell befinden, zu *verstehen*. Ein solches Verständnis kann erlangt werden, indem die einzelnen Sensoren, die dem System zur Verfügung stehen, als *Kontextinformationen* interpretiert und genutzt werden. Diese Kontextinformationen können von Entwicklern solcher Systeme verwendet werden, um zu definieren, welche konkreten Ausprägungen von Kontextinformationen (also welche gemessenen Sensorwerte), auf eine bestimmte Situation und damit auf eine bestimmte Nutzerintention schließen lassen.

Dabei fällt auf, dass die Definition solcher Systeme und der damit verbundenen Abbildung von Kontextinformationen auf Nutzerintentionen aus zweierlei Gründen nicht ausschließlich von Entwicklern solcher Systeme geleistet werden kann. Der erste Grund ist die mit diesen Systemen einhergehende Komplexität. In aktuellen Fahrzeugen sind ca. 400 Sensoren verbaut, die den Zustand des Fahrzeugs, des Fahrers und deren Umwelt erfassen. Die dabei entstehende Menge an möglichen Datenkombinationen auf Nutzerintentionen abzubilden, ist ein hochgestecktes Ziel. Der zweite, entscheidendere Grund ist jedoch, dass Nutzer bei der Bedienung ihrer FIS, individuellen Mustern folgen. Dieser Umstand macht es für einen Software-Entwickler unmöglich, ein FIS zu entwerfen, indem er sämtliche Abbildungen zwischen Kontextinformationen und Nutzerintentionen vorgibt. Vielmehr muss das System von ihm in die Lage versetzt werden, das Systemverhalten an den jeweiligen Nutzer zu adaptieren und dessen Verhalten zu *lernen*.

Mit dem Ziel, prototypische FIS erlebbar zu machen, welche die Fähigkeiten besitzen, die aktuelle Situation des Nutzers zu verstehen und auf Basis dieses Verständnisses, die Intentionen eines Nutzers zu erkennen und zu lernen, stellt sich die Frage, wie und mit Hilfe welcher technischen Konzepte, sich solche Systeme entwickeln und ausführen lassen. Motiviert durch diese Fragestellung, entstand die vorliegende Dissertation in der Abteilung FIS in der Konzernforschung der Volkswagen AG. Dazu stand nicht im Fokus, einzelne konkrete Anwendungsfälle im Hinblick auf die Nutzerakzeptanz zu evaluieren. Vielmehr konzentriert sich die vorliegende Arbeit darauf aufzuzeigen, wie solche Systeme prototypisch entwickelt werden können.

Um diese Fragestellung beantworten zu können, ist diese Arbeit wie folgt strukturiert: Kapitel 2 vermittelt zunächst Grundlagen, die zum Verständnis der darauf folgenden Kapitel erforderlich sind. In Kapitel 3 werden anschließend die Ergebnisse der im Rahmen dieser Forschungsarbeit durchgeführten Anforderungsanalyse vorgestellt, welche auf Basis von Experteninterviews durchgeführt wurde und darlegt, welche Anforderungen an Systeme zum Verstehen der aktuellen Situation und zur Erkennung von Nutzerintentionen gestellt werden. Auf Basis dieser Anforderungen werden in Kapitel 4 verschiedene Techniken zur Modellierung von Systemen zur Verarbeitung von Kontextinformationen vorgestellt und miteinander verglichen. Da dabei gezeigt werden kann, dass Ontologien eine vielversprechende Grundlage für die Modellierung

von Kontextinformationen sein können, wird im weiteren Verlauf dieses Kapitels erläutert, wie genau Ontologien dazu verwendet werden können und welche Erweiterungen dazu im Rahmen dieser Arbeit entwickelt und eingeführt wurden. Kapitel 5 wird anschließend analog dazu aufzeigen, welche Techniken zur Modellierung von Nutzerintentionen zur Verfügung stehen und wie Bayes'sche Netz dazu verwendet werden können. In Kapitel 6 wird präsentiert, wie die beiden, bis dahin vorgestellten, Teilsysteme zur Verarbeitung von Kontextinformationen und zur Erkennung von Nutzerintentionen, in einem System zusammengeführt werden können und wie dieses System implementiert werden kann. Kapitel 7 stellt das kontext- und intentionssensitive FIS *INIS* vor, welches die in der vorliegenden Dissertation erforschten und implementierten Modelle und Software-Architekturen nutzt, um die Intentionen seines Nutzers auf Basis der aktuellen Situation zu erkennen und zu lernen. Kapitel 8 fasst die Ergebnisse abschließend zusammen und gibt einen Ausblick auf weiterführende Arbeiten.

2. Grundlagen

„Die akademische Freiheit ist die Freiheit,
so viel lernen zu dürfen, wie man nur will.“

(Rudolf Virchow, deutscher Pathologe)

In diesem Kapitel werden grundlegende Informationen und Forschungsergebnisse präsentiert, die zum weiteren Verständnis der vorliegenden Arbeit nötig sind. Dazu wird in Abschnitt 2.1 zunächst auf die Merkmale aktuell zur Verfügung stehender Daten im Fahrzeug eingegangen, um ein Verständnis über die potentielle Aussagekraft dieser Daten zu schaffen. Anschließend werden in Abschnitt 2.2 Grundlagen zur Entwicklung und Modellierung kontextsensitiver Systeme erläutert, um den im später Verlauf dieser Arbeit vorgestellten Prozess zur Modellierung von Kontextinformationen einordnen und bewerten zu können. Abschließend werden in Abschnitt 2.3 Grundlagen zur Erkennung von Nutzerintentionen, dafür notwendige Softwaresysteme und deren Entwicklung und Modellierung vorgestellt, um den zweiten wissenschaftlichen Beitrag der vorliegenden Arbeit, der Modellierung von Systemen zur Erkennung Nutzerintentionen, im weiteren Verlauf einordnen zu können. Abschnitt 2.4 fasst das Kapitel abschließend zusammen.

2.1 Daten im Fahrzeug

Aktuelle Fahrzeuge verfügen über eine Vielzahl an Sensoren, die verschiedene Gegebenheiten erfassen und in elektrische Signale umwandeln [Rei10]. In diesem Abschnitt wird aufgezeigt, wie die Vielfalt an Daten im Fahrzeug strukturiert werden können und welche Techniken zur Kommunikation von Daten innerhalb eines Fahrzeuges zur Verfügung stehen. Dabei wird auch gezeigt, welche Vorarbeiten diesbezüglich in der Konzernforschung der Volkswagen AG geleistet wurden, da diese im späteren Verlauf der vorliegenden Arbeit von Bedeutung sein werden.

2.1.1 Fahrer, Fahrzeug und Umwelt

In der Literatur zeigt sich, dass die im Fahrzeug anfallenden Daten, ihrem Ursprung entsprechend, in die Domänen *Fahrer*, *Fahrzeug* und *Umwelt* eingeteilt werden können und es je nach Anwendungsfall auch Wechselwirkungen zwischen den einzelnen Daten dieser Domänen geben kann. Die Gesamtheit der Daten aus diesen drei Domänen wird dabei als *Kontext* referenziert. Die folgenden Zitate und Erklärungen bestätigen diese Aussagen exemplarisch.

Beim Entwurf eines Systems zum Kontextmanagement-Systems sagt Hoch in seiner Dissertation [Hoc09], dass zur vollständigen Erfassung des aktuellen Nutzerkontexts möglichst viele, vor allem aber Daten aus den Bereichen Fahrer, Fahrzeug und Umwelt berücksichtigt werden müssen.

> *„Der Kontext wird in dieser Arbeit nicht applikationsbezogen definiert, sondern liefert in Form der Kontextparameter eine allumfassende, einheitliche Wissensbasis zur Situationsbeschreibung für eine Entität eines Fahrzeug-Umwelt-Fahrer-Systems.“*
>
> (Hoch [Hoc09])

Im *Handbuch Fahrerassistenz* [WHW09] stellen Winner et al. fest, dass es bei der Benutzung und auch der Bewertung von Fahrerassistenzsystemen nicht genügt lediglich auf die reine Funktion des Assistenzsystems zu schauen, sondern, dass das Zusammenspiel zwischen Fahrer, Fahrzeug und Umwelt (bzw. Umgebung) betrachtet werden müssen.

> *„Es wird deutlich, dass es sich hierbei um eine Zieldimension handelt, die nur in einer Gesamtfahrzeug-Umgebung bewertet werden kann – es muss immer das Gesamtsystem aus Fahrer, Fahrzeug und Umgebung betrachtet werden.“*
>
> (Winner et al. [WHW09])

Bei der Entwicklung eines *Intelligent Driver Assistance Systems (I-DAS)*, merken Kannan et al. [KTK10] an, dass es zur Entwicklung eines intelligenten Systems entscheidend ist, neben den physikalischen Eigenschaften des Fahrzeugs, auch die aktuelle Fahrsituation, Fahreraktivitäten und die Fahrzeugumgebung zu berücksichtigen.

> *„The inputs are taken from the driver, vehicle, and external environment sensors and given to the system.“*
>
> (Kannan et al. [KTK10])

Auch Knoll schreibt, dass bei der Entwicklung eines Fahrerassistenzsystems (FAS) sowohl das Fahrzeug und dessen Umfeld, als auch der Fahrer selbst berücksichtigt werden müssen. Nur wenn Daten aus allen drei Domänen berücksichtigt werden, so Knoll weiter, “[...] sind eine Verbesserung des Komforts, der Verkehrssicherheit, und letztendlich die Bereitschaft zum Kauf zu erwarten. „[Kno10].

> *„Fahrer, Fahrzeug mit Fahrerassistenzsystem (FAS) und das Umfeld des Fahrzeugs wirken in Raum und Zeit eng zusammen.“*
>
> (Knoll [Kno10])

Schäuffele et al. stellen fest, dass sowohl der Fahrer, als auch die Umwelt das Fahrzeug beeinflussen können und die drei Komponenten Fahrer, Fahrzeug und Umwelt daher ein übergeordnetes Gesamtsystem bilden. Die Autoren berichten weiter, dass "[...] zwischen den Komponenten Fahrer, Fahrzeug und Umwelt zahlreiche Signalflüsse bestehen.„[SZ13].

> *„Fahrer und Umwelt können das Verhalten des Fahrzeugs beeinflussen und sind Komponenten des übergeordneten Systems Fahrer-Fahrzeug-Umwelt.“*
>
> (Schäuffele et al. [SZ13])

Auch in den Arbeiten von Fletcher [FPZ05] und Zhang [ZWH04] lässt sich, wenngleich auch nicht explizit erwähnt, erkennen, dass von einer solchen Klassifizierung ausgegangen wird. Eine Einteilung, der in einem Fahrzeug anfallenden Daten, in die Kategorien Fahrer, Fahrzeug und Umwelt wird aufgrund der oben beschriebenen, aktuellen Forschungslage und der in der Volkswagen AG gesammelten Erfahrung mit dem Umgang von Daten im Automobil, auch im weiteren Verlauf dieser Arbeit Anwendung finden.

2.1.2 Kommunikation von Daten

Die Kommunikation von Daten im Fahrzeug, wird seit den 1980er Jahren überwiegend mit Hilfe eines Controller Area Network (CAN) abgewickelt [MS10]. Die Teilnehmer eines solchen CAN-Netzwerkes sind dabei durch ein Bus-System miteinander verbunden, so dass alle auf den Bus geschriebenen Informationen allen anderen Teilnehmern zur Verfügung stehen. Über das Aussenden von Botschaften, die mit einer bestimmten ID versehen werden, können so Informationen an alle anderen Teilnehmer verteilt werden. Eine Komponente, die beispielsweise die Fahrzeuggeschwindigkeit erfasst, würde diese zyklisch auf dem CAN-Bus zur Verfügung stellen. Die Kommunikation kann daher auch als *signalbasiert* beschrieben werden und findet in einer n-zu-n Beziehung statt.

Im Laufe der darauf folgenden Jahre, wurden aufgrund wachsender Anforderungen und eines steigenden Kostendrucks, weitere Kommunikationskanäle entwickelt und in Fahrzeuge integriert. Dazu zählen beispielsweise das Local Interconnect Network (LIN) [ZS14], Media Oriented Systems Transport (MOST) [ZS14], FlexRay [Rau08, ZS14] oder auch BroadR Reach[1] [HSM12].

2.1.2.1 Abstraktion durch EXLAP

Aufgrund der dadurch entstehenden Vielfalt von Kommunikationskanälen, wurde bereits im Jahr 2006 in der Konzernforschung der Volkswagen AG damit begonnen das eXtensible Lightweight Asynchronous Protocol (EXLAP)[2] zu entwickeln. Mit EXLAP wurde insbesondere das Ziel verfolgt, die Vielzahl bestehender Kommunikationskanäle so zu abstrahieren, dass nachgelagerte Komponenten keine Kenntnis darüber haben müssen, über welchen Kanal sie Botschaften empfangen oder versenden müssen.

[1] BroadR Reach wird von der Open Alliance entwickelt. Weitere Informationen können der Website der Open Alliance entnommen werden: http://www.opensig.org/. Zuletzt überprüft am 19.10.2016.

[2] EXLAP wurde unter der *Creative Commons Namensnennung-Weitergabe unter gleichen Bedingungen 3.0 Deutschland Lizenz* veröffentlicht unter kann unter folgendem Link abgerufen werden: http://tinyurl.com/exlap-vw. Zuletzt überprüft am 19.10.2016.

EXLAP definiert dabei die Kommunikation oberhalb der Transport-Schicht (vgl. OSI-Modell [Sie93]). In bis dato durchgeführten Forschungsprojekten kamen bei den darunter liegenden Schichten in überwiegender Mehrzahl TCP, IP und Ethernet zum Einsatz.

```
<Profile name="Math" version="1.1" xmlns="http://exlap.de/v1_2/protocol">
   <About language="en">A simple mathematical service</About>
   <Object context="global" url="Statistics" characteristic="dynamic">
      <Description language="en">Object that represents the statistics of the
          service operations.</Description>
      <!-- @param TotalSum The total sum of all operation results. -->
      <Absolute name="TotalSum" unit="1"/>
      <!-- @param OperationsCount The number of performed operations. -->
      <Absolute name="OperationsCount" unit="1" resolution="1"/>
   </Object>
   <Function url="Add">
      <Description language="en">Simple function to add two values.</Description>
      <In>
         <!-- @param SummandA The first summand -->
         <Absolute name="SummandA" unit="1"/>
         <!-- @param SummandB The second summand -->
         <Absolute name="SummandB" unit="1"/>
      </In>
      <Out>
         <!-- @param Sum The resulting sum (summand a + summand b) -->
         <Absolute name="Sum" unit="1"/>
         <!-- @param Result Result status of the operation -->
         <Enumeration name="Result">
            <!--  @enum ok The operation was successful -->
            <Member id="ok"/>
            <!--  @enum error An error has occurred during the operation -->
            <Member id="error"/>
         </Enumeration>
      </Out>
   </Function>
   <Function url="Div">
      <Description language="en">Simple function to divide two values.</
          Description>
      <In>
         <!-- @param Divident The divident -->
         <Absolute name="Divident" unit="1"/>
         <!-- @param Divisor The divisor -->
         <Absolute name="Divisor" unit="1"/>
      </In>
      <Out>
         <!-- @param Quotient The division result (i.e. the quotient) -->
         <Absolute name="Quotient" unit="1"/>
         <!-- @param Result Result status of the operation -->
         <Enumeration name="Result">
            <!--  @enum ok The operation was successful -->
            <Member id="ok"/>
            <!--  @enum divisionbyzero A division by zero error has happened -->
            <Member id="divisionByZero"/>
            <!--  @enum error An other error has happened -->
            <Member id="error"/>
         </Enumeration>
      </Out>
   </Function>
</Profile>
```

Quelltext 2.1: Exemplarisches Exlap-Interface zur Definition eines Exlap-Dienstes

Auch das Kommunikationsparadigma wandelte sich mit EXLAP. Statt einer signalbasierten n-zu-n-Kommunikation, setzt EXLAP auf eine servicebasierte 1-zu-n-Kommunikation, bei der ein *EXLAP-Server* sowohl Informationen als auch Methoden, für eine unbestimmte Anzahl an *EXLAP-Clients*, zur Verfügung stellt. Für das bereits oben genannte Beispiel der Fahrzeuggeschwindigkeit würde dies bedeuten, dass EXLAP-Server zunächst das *Datenobjekt* Fahrzeuggeschwindigkeit bereitstellt. Der

dazugehörige EXLAP-Client kann sich bei diesem EXLAP-Server anschließend auf Änderungen dieses Objekts registrieren und wird von diesem fortan über Änderungen der Fahrzeuggeschwindigkeit informiert.

Daraus ergibt sich, dass während der Entwicklung eines EXLAP-Clients die von einem EXLAP-Server zur Verfügung gestellten Daten und Methoden bekannt sein müssen. Bereits während der Entwicklung des EXLAP-Servers liegt der Schwerpunkt auf der Definition angebotenen Datenobjekte und Funktionen. Quelltext 2.1 zeigt exemplarisch anhand eines Dienstes, der mathematischen Operationen zur Verfügung stellt, wie EXLAP-Dienste mit Hilfe einer XML-Datei definiert werden können. Dabei können neben einer Beschreibung des Dienstes (vgl. Element `About` in Zeile 2), auch Datenobjekte (vgl. Element `Object` in Zeile 3ff.) und Funktionen (vgl. Elemente `Function` in Zeile 10ff. und 30ff.) festgelegt werden. Ausgehend von dieser Definition, können dann die Grundgerüste der Quellcodes für EXLAP-Server und EXLAP-Clients generiert werden, so dass ein Entwickler bei der Verwendung keinerlei Aufwand in die Implementierung der Kommunikationsabläufe zwischen Server und Client investieren muss, sondern sich einzig und allein auf die Ausgestaltung der eigentlichen Funktionen konzentrieren kann.

EXLAP wird insbesondere in Kapitel 6 der vorliegenden Arbeit eine erneute Erwähnung finden, wenn erläutert wird, wie die bis dahin vorgestellten Techniken und Methoden implementiert wurden.

2.1.2.2 Die OSGi Service Plattform

Während EXLAP dabei hilft, Daten zwischen verschiedenen Recheneinheiten innerhalb eines Fahrzeuges zu kommunizieren, ist dessen Einsatz zur Kommunikation zwischen Software-Komponenten, die auf einer Hardware zum Einsatz kommen, schon allein wegen der Tatsache, dass jede Kommunikation mittels TCP geschieht und daher den Netzwerkstacks des Betriebssystems durchlaufen muss, nicht optimal geeignet. Ein Plattform-Framework, das für solche Anwendungsfälle ebenfalls bereits vor Beginn der in dieser Arbeit beschriebenen Forschungsaktivitäten zur prototypischen Umsetzung neuer FIS in der Konzernforschung der Volkswagen AG zum Einsatz kommt, ist die *OSGi Service Plattform*. Seit 1999 wird sie von der *Open Service Gateway Initiative* (seit 2004 bekannt als *OSGi Alliance*) als Zusammenschluss mehrerer Unternehmen, wie Oracle oder IBM entwickelt [Wüt08, Lüd12] und ermöglicht es, ein Java-Software-System durch standardisierte Schnittstellen zu modularisieren und die Ausführung dieser einzelnen Module zentral zu überwachen.

Zentraler Bestandteiler der OSGi Service Platform ist das *OSGi-Framework*. Es schafft einen Ausführungsrahmen für die einzelnen Module – die sogenannten *OSGi-Bundles* – der es erlaubt, diese kontrolliert durch ihren Lebenszyklus zu führen. Die möglichen Zustände in diesem Zyklus sind in Abbildung 2.1 dargestellt. Sobald ein OSGi-Bundle (repräsentiert durch eine `.jar`-Datei) in den Hauptspeicher des ausführenden Systems geladen wurde, befindet es sich im Zustand `installed`. Anschließend wird das OSGi-Framework versuchen, dieses Bundle *aufzulösen*. Dazu werden Abhängigkeiten zu anderen OSGi-Bundles erkannt und, falls erforderlich, diese anderen Bundles in den Zustand gebracht, in dem sie die benötigten Klassen zur Verfügung stellen können. Kann das Framework alle Abhängigkeiten auflösen, befindet sich das Bundle im Zustand `resolved` und kann seine Klassen, in Abhängigkeit der implementierten

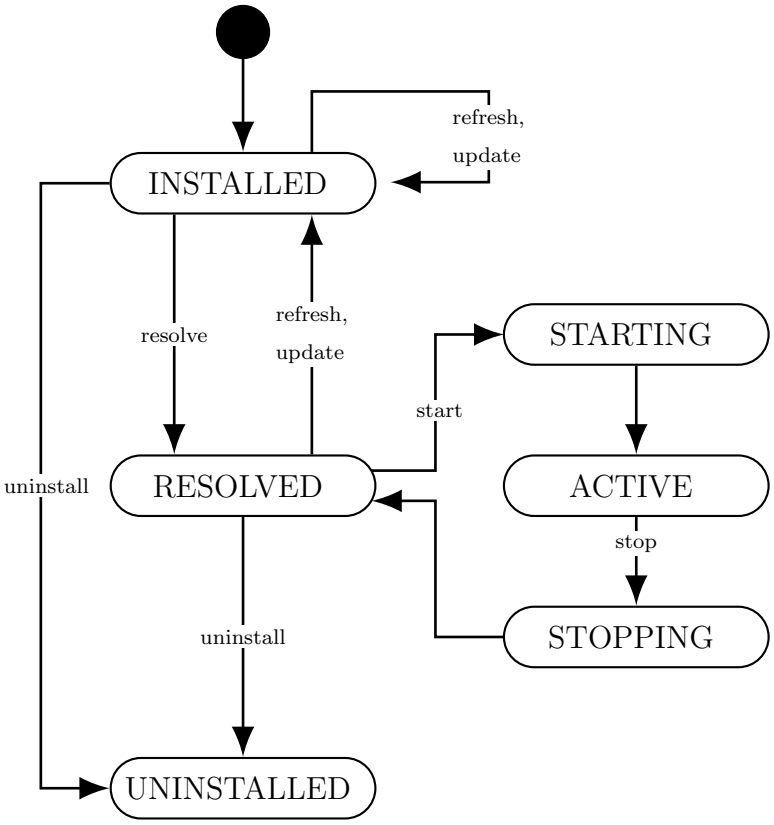

Abbildung 2.1: Lebenszyklus eines OSGi-Bundles als Zustandsautomat (entnommen aus [Wüt08])

Sichtbarkeit, für andere OSGi-Bundles zur Verfügung stellen. Zur tatsächlichen Ausführung von Quellcode des Bundles ist es bis dahin jedoch noch nicht gekommen. Dies geschieht erst, wenn das Framework im nächsten Schritt das OSGi-Bundle startet und es dadurch in den Zustand `starting` bzw. `active` überführt. Durch eine definierte Einsprungmethode (die `start`-Methode), wird so eine bestimmte Methode des OSGi-Bundles vom Framework aufgerufen. Ausgehend von dieser Methode, kann der Entwickler den weiteren Ablauf und die Funktionalität des OSGi-Bundles kontrollieren und steuern. Wird das OSGi-Framework heruntergefahren, beendet es zunächst all seine aktiven Bundles. Dazu wird diesen vom Framework die Möglichkeit gegeben eigene Aktivitäten zu beenden oder Ressourcen wieder freizugeben, indem erneut ein standardisierte Methode (die `stop`-Methode) aufgerufen wird. Während diese Methode ausgeführt wird, befindet sich das Bundle im Zustand `stopping`. Nach Abschluss der Methode wieder im Zustand `resolved`. Abschließend können Bundles deinstalliert werden (Zustand `uninstalled`), wodurch ihre Klasse dem OSGi-Framework und anderen Bundles nicht länger zur Verfügung stehen.

Zur Kommunikation zwischen OSGi-Bundles, haben Entwickler die Möglichkeit sogenannten *OSGi-Services* am Framework zu registrieren. So wird es anderen Bundles (oder Services) möglich, bestimmte Funktionsumfänge oder Daten, anderen Bundles zur Verfügung zu stellen. Das für die Kommunikation zugrundeliegende Entwurfsmuster (engl. design pattern), wird als *Whiteboard Pattern* bezeichnet [KH04]. Zur Erklärung dieses Musters sei exemplarisch angenommen, dass ein OSGi-Service im Framework gestartet wurde, der die aktuelle Fahrzeuggeschwindigkeit zur Verfügung

```
public Interface VehicleSpeedListenner CMData {

    public void receiveVehicleSpeed(double currentVehicleSpeed);

}
```

Quelltext 2.2: Exemplarisches Listener-Interface zum Empfangen einer Fahrzeuggeschwindigkeit

stellen kann. Dieser Service soll wahlweise zyklisch, oder bei Änderung der Fahrzeuggeschwindigkeit, weiteren Bundles im Framework die aktuelle Geschwindigkeit des Fahrzeugs mitteilen. Aufgrund der hohen Dynamiken im Framework, kennt der Service jedoch die Bundles, welche die Fahrzeuggeschwindigkeit empfangen sollen, nicht konkret. Während der Entwicklung wurde das in Quelltext 2.2 dargestellte *Listener-Schnittstelle* entworfen, welches eine Methode zum Empfangen einer Fahrzeuggeschwindigkeit beinhalten. Bundles, welche die aktuelle Fahrzeuggeschwindigkeit zur Laufzeit von besagtem Service empfangen sollen, implementieren diese Schnittstelle und registrieren sich damit bei der sogenannten *Service-Registry*. Soll der Service, welcher die Fahrzeuggeschwindigkeit kennt, diese nun weitergeben, so hat er die Möglichkeit bei der Service-Registry die Referenzen auf die Bundles zu bekommen, die sich als Listener für die Fahrzeuggeschwindigkeit registriert haben. Die anschließende Kommunikation findet direkt und in einer 1-zu-1-Beziehung zwischen den Services statt. Das Aufbauen dieser Beziehung geschieht, wie beschrieben, über die Service-Registry.

Auch die OSGi Service Platform, sowie das dazugehörige Whiteboard Pattern wird in Kapitel 6 erneut eine zentrale Rolle spielen, da es als Basis für die im Rahmen dieser Arbeit durchgeführten Implementierungen diente.

2.2 Kontext & kontextsensitive Systeme

Zu Beginn der 1990er Jahre setzen sich Wissenschaftler intensiver mit der Fragestellung auseinander, wie Computersysteme entwickelt werden können, die den aktuellen Kontext ihrer Verwendung berücksichtigen. Dabei lag der Fokus zunächst insbesondere auf mobilen Systemen. So entwarfen Schilit et al. [ST94] etwa die sog. *Active Maps*, die es erlaubten, mit Personen und Objekten in der näheren Umgebung zu interagieren und das eigene Applikationsverhalten in Abhängigkeit zum jeweiligen Aufenthaltsort anzupassen. Etwas generischer wird die Definition von Kontext (und damit auch von kontextsensitiven Systemen) Ende der 1990er Jahre, zum Beispiel mit der Definition von Dey et al.:

> *„Context is any information that can be used to characterize the situation of an entity. An entity is a person, place, or object that is considered relevant to the interaction between a user and an application, including the user and applications themselves.“*
>
> (Dey et al. [DA99])

Als Kontext werden also Informationen verstanden, die die Situation, einer für die Interaktion zwischen Nutzer und Applikation relevanten Entität, näher beschreiben.

Dazu zählen nicht zuletzt auch der Nutzer und die Applikation selbst. Aufgrund der Vielzahl der zur Bestimmung der aktuellen Situation erforderlichen Informationen, ist die Erstellung kontextsensitiver Systeme keine triviale Aufgabe. Daher wurde, ebenfalls beginnend in den 1990er Jahren, damit begonnen, Techniken und Methoden zu untersuchen und zu entwickeln, die es erlauben kontextsensitive Systeme modellbasiert zu entwickeln. Die dabei entstandenen Ansätze werden in den folgenden Abschnitten vorgestellt.

2.2.1 Schlüssel-Wert-Paare

Ein Schlüssel-Wert-Paar bietet eine sehr primitive Möglichkeit, Werten eine bestimmte Semantik zu verleihen, indem sie einem bestimmten Schlüssel zugeordnet werden. Im Rahmen kontextsensitiver Systeme können so, durch Sensoren ermittelte Werte, einem bestimmten Schlüssel zugeordnet werden. Je nach tatsächlicher Ausprägung des Wertes eines bestimmten Schlüssels kann sich das kontextsensitive System dann ensprechend adaptieren. Wie Chen und Kotz zeigen [CK00], wurden Schlüssel-Wert-Paare vor allem in den 1990er Jahren verwendet um Systeme zu modellieren, die ihr Verhalten in Relation zum aktuellen Aufenthaltsort des Nutzers verändern [STW93, SAW94, VB96, Maa98]. Trotz aller Einfachheit dieser Modelle, können einfache kontextsensitive Systeme damit schnell und ohne zu große Einstiegshürden erstellt werden. Für komplexere Systeme mangelt es Schlüssel-Wert-Paaren jedoch an Ausdrucksstärke. So gibt es beispielsweise keine Möglichkeiten, Abhängigkeiten und Beziehungen zwischen einzelnen Kontextinformationen zu modellieren.

2.2.2 Logikbasierte Modelle

Einen historischen Abriss über die Entwicklung Logikbasierter Modelle von Kontextinformationen wurde 2004 von Strang und Linnhoff-Popien veröffentlicht [SLP04]. Zusammenfassend lässt sich sagen, dass Logikbasierte Modelle Regeln verwenden (zum Beispiel mit Hilfe der Prädikatenlogik), um durch Sensoren gemessene Kontextinformationen, mit einer Semantik zu versehen. Die Verwendung Logikbasierter Modelle zur Beschreibung von Kontextinformationen geht zurück auf John McCarthy, der als einer der ersten danach strebte, Kontextinformationen zu formalisieren, indem er sie als sogenannte *First-Class-Objekte*[3] betrachtete und dadurch eine Axiomatisierung[4] der Kontextinformationen ermöglichte [McC93, MB97]. Akman und Surav basieren ihre Arbeit auf der von Barwise und Perry eingeführten *Situation Theory* [BP81] und erweitern diese, so dass Situationen, die durch Kontextinformationen beschrieben werden, als First-Class-Objekte beschrieben werden können [AS97]. Auch Gray et al. verwenden Logikbasierte Modelle zur Beschreibungen von Kontextinformationen, stellen zusätzlich dazu auch fest, dass Kontextinformationen, die zu meist durch Sensoren ermitteln werden, immer einer gewissen Ungenauigkeit unterliegen [GS01]. Dieser Umstand wird im weiteren Verlauf dieser Arbeit von Bedeutung sein.

[3] In Abhängigkeit zu einer konkreten Programmiersprache, sind First-Class-Objekte jene, die als Parameter oder Ergebnis einer Methode eingesetzt oder Variablen zugewiesen werden können [Sco09].

[4] Axiomatisierung erlaubt es, ausgehend von einer Grundmenge an Fakten (den Axiomen), durch Deduktion auf andere Fakten zu schließen [Car68].

Giunchiglia und Ghidini beschäftigen sich in ihren Arbeiten mit dem Auswerten Logikbasierter Modelle [Giu92, GG01], sodass durch Logikbasierte Modelle modellierte kontextsensitive Systeme zur Laufzeit ausgewertet werden können und eine Situationsbeschreibung weiteren Software-Komponenten zur Verfügung gestellt werden können. Durch die Verknüpfung und Kaskadierung der so erstellen Regeln, können Logikbasierte Modelle komplexere Situationen einfacher beschreiben als einfache Schlüssel-Wert-Paare. Was jedoch auch Logikbasierten Modellen fehlt, ist die Fähigkeit, Hierarchien zwischen einzelnen Kontextinformationen abzubilden [SLP04].

2.2.3 Profilbasierte Modelle

Profilbasierte Modelle beschreiben in erster Linie, in welcher Struktur Kontextinformationen zur Laufzeit vorliegen werden. Im Vergleich zu den bisher vorgestellten Modellen erlauben sie es auch, Hierarchien und Abhängigkeiten zwischen verschiedenen Kontextinformationen abzubilden [SLP04]. Ein frühes Beispiel für Profilbasierte Modelle für Kontextinformationen stellen die *stick-e-notes* von Brown et al. dar [BBX97]. Sie nutzen die Standard Generalized Markup Language (SGML) und entwickeln eine eigene Document Type Definition (DTD), die festlegt, welche Kontextinformationen in welchem Umfang das Verhalten des Systems zur Laufzeit verändern.

Weitere Profilbasierte Modellierungen von Kontextinformationen, wie zum Beispiel Composite Capabilities/Preference Profiles (CC/PP)[5], basieren auf dem Resource Description Framework (RDF)[6]. CC/PP wurde ursprünglich zur Beschreibung der Fähigkeiten eines bestimmten Geräts (zum Beispiel einem Mobiltelefon) gegenüber einem weiteren Gerät (zum Beispiel einem Webserver) entwickelt. CC/PP unterliegt sowohl einem festgelegtem Vokabular, als auch einer festgelegten Hierarchie [SLP04]. Dynamisches Hinzufügen neuer Informationen oder neuer Strukturen ist daher unter Verwendung von CC/PP nicht möglich.

Es existieren auch Ansätze, CC/PP als allgemeine Beschreibung von Kontextinformationen zu erweitern, so zum Beispiel die *Comprehensive Structured Context Profiles* (CSCP) [HBSS02] oder die *CC/PP Context Extension* [IRRH03]. Zwar können im Vergleich zu CC/PP Hierarchien flexibel zur Modellierungszeit festgelegt werden und sind nicht, wie bei CC/PP, durch die Modellierungssprache selbst vorgegeben, jedoch fehlt beiden Ansätzen die Möglichkeit, Beziehungen zwischen Kontextinformationen auszudrücken und Kontextinformationen mit einer Wahrscheinlichkeit zu versehen.

Das *Reflective Conceptual Model* von Capra et al. [CEM01] ermöglicht es, einzelnen Applikationen eines Gesamtsystems XML-basierte Profile zu erstellen, in denen definiert wird, welche Auswirkungen bestimmte Veränderung am Kontext des Gesamtsystems auf eben diese einzelne Applikation haben. Quelltext 2.3 zeigt exemplarisch wie die Definition solcher Profile durchgeführt werden kann. In diesem Beispiel trennt ein mobiles Gerät die Verbindung zu einem Server, wenn der Batterieladestand unter einen bestimmten Wert `x` fällt.

[5] Spezifikation des World Wide Web Consortiums: http://www.w3.org/Mobile/CCPP/. Zuletzt überprüft am 19.10.2016.

[6] Spezifikation des World Wide Web Consortiums: http://www.w3.org/TR/REC-rdf-syntax/. Zuletzt überprüft am 19.10.2016.

```
<RESOURCE name="battery">
    <STATUS operator="lessEqual" value=x/>
    <BEHAVIOUR policy="disconnect"/>
</RESOURCE>
```

Quelltext 2.3: Reflective Conceptual Model (übernommen aus [CEM01])

Im Vergleich zu Schlüssel-Wert-Paaren bieten Profilbasierte Modelle den Vorteil, dass sich mit Ihnen sowohl eine Situationsbeschreibung, als auch die Auswertung einzelner Kontextinformationen vornehmen lässt. Nachteilig ist jedoch zu erwähnen, dass sowohl Indulska et al. [IRRH03] als auch Butler [But02] Probleme mit profilbasierten Modellen aufgezeigt haben. Zurückzuführen lassen sich diese Probleme nach Strang et al. auch auf die Limitierungen der Extensible Markup Language (XML) und RDF [SLP04]. Strang et al. berichten auch, dass geforderte Anforderungen an Kontextmodelle von profilbasierten Modellen unerfüllt bleiben [SLP04]. Dazu zählen unter anderem, dass profilbasierte Modelle keine automatische Überprüfung bezüglich Vollständigkeit und Widerspruchsfreiheit bereitstellen. Da solche Überprüfungen nicht Teil der Modellierungssprache sind, müssen sie individuell auf Anwendungsebene durchgeführt werden. Es fällt auch auf, dass bestehende profilbasierte Modelle keine Möglichkeiten bieten Aussagen über die Wahrscheinlichkeit von einzelnen Kontextinformationen zu integrieren.

2.2.4 Grafische Modelle

Grafische Modelle bieten in der Regel den Vorteil einer einfachen und verständlichen Art und Weise der Modellierung. Eine der wohl verbreitesten grafischen Modellierungssprachen – die Unified Modeling Language (UML)[7] – dient als Grundlage der Ansätze von Bauer [Bau03] und Sheng [SB05]. Bauer zeigt, dass sich Kontext auf Basis von UML-Klassendiagrammen modellieren lässt, um im Bereich der Flugsicherung die aktuelle Situation von Flugzeugen zu erfassen.

Henricksen et al. merken hingegen an, dass UML zur Modellierung bestimmter Eigenschaften von Kontextinformationen ungeeignet sei [HIR03]. Dazu zählen unter anderem Eigenschaften wie, Historien (also das Abbilden eines Werteverlaufs für bestimmte Kontextinformationen), Unsicherheiten, Unvollständigkeiten, Abhängigkeiten sowie die Diversität von Kontextinformationen. Es existieren zu dieser Zeit zwar Ansätze, die diese Anforderungen partiell erfüllen (beispielsweise Ansätze zur Modellierung von Datenqualitäten [WRK95] sowie eine Vielzahl von Techniken zu temporalen Erweiterungen von Entity-Relationship-Modellen [HC99]), jedoch existieren laut Henricksen et al. keine Ansätze, die alle Anforderungen gleichzeitig erfüllen [HIR03]. Anstelle der Verwendung von UML schlagen Henricksen et al. daher einen Ansatz vor, der auf der Verwendung des sogenannten Object-Role Modelings (ORM) [Hal01] basiert und als Context Modeling Language (CML) bezeichnet wird [HIR02, HIR03]. Das zentrale Mittel zur Modellierung mittels ORM sind *Fakten*. Henricksen erweiterte das ORM um die Möglichkeit, verschiedene Fakten zu kategorisieren. Durch eine solche Kategorisierung wird es beispielsweise möglich,7 statische Fakten von dynamischen, sich zur Laufzeit verändernden Fakten, zu unterscheiden,

[7] UML Spezifikationen: http://www.omg.org/spec/UML/. Zuletzt überprüft am 19.10.2016.

oder die Quelle eines Faktes abzubilden. Des Weiteren führten Henricksen et al. die Möglichkeit ein, Fakten und deren Beziehungen mit einer zeitlichen Komponente zu versehen, um Historien von Kontextinformationen abbilden zu können. Abschließend sei noch erwähnt, das Henricksen et al. ORM so erweiterten, dass Änderungen in einer Abhängigkeit zwischen zwei Fakten automatisch zu einer Veränderung einer anderen Abhängigkeit führen können.

2.2.5 Objektorientierte Modelle

Die *objektorientierte Modellierung* von Kontextinformationen basiert auf dem weit verbreiteten Konzept der objektorientierten Programmierung. Die übernommen Konzepte, wie die Möglichkeiten zur Vererbung, zur Kapselung und zur Wiederverwendung, unterstützen den Modellierer dabei, Kontextinformationen in Form von Klassen und Schnittstellen abzubilden [SLP04]. Beziehungen zwischen Kontextinformationen können als Verweise zwischen Klassen abgebildet werden [Zim07].

Eine erste Verwendung einer objektorientierten Modellierung von Kontextinformationen lässt sich in der Arbeit von Bouzy et al. finden [BC97], die eine Modellierung von Kontextinformationen nutzten, um Computer in die Lage zu versetzen das ca. 2.500 Jahre alte Brettspiel *Go*[8] spielen zu können. Der für Go entscheidende Kontext kann zeit-, ziel-, und umgebungsorientiert sein und global modelliert werden. Bouzy et al. zufolge ist für den Erfolg des Go-spielenden Programms ausschlaggebend, dass es die verschiedenen Kontexte berücksichtigt. So sollte sich das Programm beispielsweise zu Beginn des Spiels (zeitorientiert) andere Ziele verfolgen (zielorientiert) als im späteren Verlauf des Spiels.

Ein weiteres Beispiel für eine objektorientierte Modellierung von Kontextinformationen, stellen laut Strang et al. [SLP04] die von Schmidt et al. eingeführten *Cues* [SBG99, Sv01] dar. Der entscheidende Unterschied zu anderen Ansätzen liegt darin, dass aus Sensoren in einem ersten Schritt auf sogenannte Cues abstrahiert wird. Schmidt et al. zufolge wird dadurch eine Unabhängigkeit zwischen der Sensorik-Hardware und den Datenverarbeitenden Softwareschichten gewährleistet. Außerdem können Cues dazu genutzt werden, um die von den Sensoren erzeugten Datenraten zu reduzieren, indem zum Beispiel nicht alle Rohdaten direkt weitergeleitet werden, sondern statistische Mittelwerte über eine bestimmte Zeit gebildet werden [Sv01].

Cheverst et al. setzten mit einer objektorientierten Modellierung von Kontextinformationen einen kontextsensitiven Reiseführer um [CMD99]. Dabei bewerten sie die Konzepte der Kapselung von Informationen und Daten, sowie die Möglichkeiten mit dynamischen Daten umgehen zu können, als hilfreich und wertvoll.

Auch Dey et al. stellen mit dem *Context Toolkit*[DAS01] eine Möglichkeit vor, Kontextinformationen durch objektorientierte Modellierung zu beschreiben. Bei der Implementierung wurden dabei Schwerpunkte auf eine verteilte, servicebasierte Kommunikation zwischen allen beteiligten Komponenten gesetzt.

Weitere Beispiele für eine objektorientierte Modellierung von Kontextinformationen lassen sich im *Hydrogen-Framework*[HSP+03] und dem *Java Context Awareness Framework* [Bar05] finden.

[8] Umfangreiche Informationen zu Go sind unter http://www.gobase.org zu finden. Zuletzt überprüft am 19.10.2016.

2.2.6 Ontologiebasierte Modelle

Ontologien können, allgemein gesprochen, genutzt werden, um Wissen einer bestimmten Domäne formal zu beschreiben [UG+96]. Die Repräsentation des Wissens einer bestimmten Domäne basiert dabei auf einer vereinfachten und abstrahierten Darstellung der abzubildenden Daten und Fakten – der *Konzeptualisierung* [Gru93, GN87]. Solche Konzepte können von der reinen Existenz, von Gegenständen und Fakten, über Eigenschaften, bis hin zu Beziehungen zwischen diesen, reichen. In der Domäne des Automobils könnten so beispielsweise die Konzepte *Kraftfahrzeug*, *Personenkraftwagen (PKW)* und *Person* modelliert werden. Um § 4 Abs. 4 des deutschen Personenbeförderungsgesetzes gerecht zu werden, kann anschließend noch modelliert werden, dass das Konzept *PKW* ein *Kraftfahrzeug* ist, das nicht mehr als neun *Personen* transportieren kann und zum Zwecke der Personenbeförderung dient. Stück für Stück kann die Realität so mit Hilfe von Ontologien formal beschrieben werden. Die dabei erstellten Konzepte, repräsentiert durch Klassen in einer Ontologie, können durch sogenannte *Data Properties* angereichert werden. Das Konzept *PKW* kann so beispielsweise die Data Property `Anzahl der Sitzplätze` zugewiesen werden, um hinterlegen zu können, wie viele Sitzplätze eine konkrete Ausprägung des Konzepts *PKW* hat. Diese konkreten Ausprägungen werden als *Individuum* bezeichnet.

Eine der Stärken von Ontologien besteht darin, implizites Wissen zur Laufzeit explizit zu machen. Dieser Prozess wird als *Reasoning* bezeichnet und wird von einem sogenannten *Reasoner* durchgeführt. Ein Individuum des Konzepts *Kraftfahrzeug* mit der Eigenschaft fünf *Personen* transportieren zu können und der Personenbeförderung zu dienen, kann so automatisch als *PKW* klassifiziert werden, auch wenn diese Tatsache nicht explizit für dieses konkrete Individuum modelliert wurde. Durch die Fähigkeit implizites Wissen explizit machen zu können, sind Reasoner auch in der Lage Widersprüche innerhalb der Ontologie aufzudecken [HKRS08, Ber14]. Diese Fähigkeit ist während der Modellierung von Fakten und deren Zusammenhänge innerhalb einer Domäne von Vorteil, da so automatisiert überprüft werden kann, ob das Modell in sich konsistent ist.

Auch wenn Ontologien in ihren Grundzügen stark an objektorientierte Programmierung erinnern, so gibt es entscheidende Unterschiede. Die erste Besonderheit ist die sogenannte *Open World Assumption* [HKRS08]. Sie besagt, dass Fakten, die nicht als Bestandteil der Ontologie modelliert wurden, nicht automatisch als ungültig betrachtet werden können. Ergänzt man das oben genannte Beispiele um das Konzept des *Lastkraftwagen (LKW)* und modelliert zusätzliche die Tatsache, dass auch ein *LKW* ein *Kraftfahrzeug* ist, wird ein Reasoner ein Individuum des Typs *Kraftfahrzeug* nicht zweifelsfrei klassifizieren können, da es sowohl *PKW* als auch *LKW* sein könnte. Erst durch das Hinzufügen des Fakts, dass ein *Kraftfahrzeug* nicht *LKW* und *PKW* gleichzeitig sein kann und das Hinzufügen der Bedingung, dass *PKW* zur Personen- und *LKW* zur Güterbeförderung eingesetzt werden, wird es einem Reasoner möglich, *Kraftfahrzeuge* zweifelsfrei als *PKW* oder *LKW* zu klassifizieren. Die zweite Besonderheit, die es bei der Verwendung von Ontologien zu beachten gilt, ist die sogenannte *Non-unique Name Assumption*. Diese Annahme geht davon aus, dass Ressourcen durch mehrere Namen identifiziert werden können [HKRS08]. Auf

Ontologien bezogen bedeutet diese Annahme, dass zum Beispiel mehrere Namen auf ein und das selbe Individuum verweisen können [Ber14].

Ontologien wurde bereits Anfang der 1990er Jahre als vielversprechend erkannt. So findet sich beispielsweise in der Arbeit von Gruber ein System zur einheitlichen Beschreibung von Ontologien, die in verschiedenen Beschreibungssprachen erstellt und bearbeitet werden können [Gru91, Gru93]. Einer der ersten Ansätze zur Modellierung von Kontextinformationen mit Hilfe von Ontologien lässt sich in der Arbeit von Öztürk und Aamodt finden [ÖA97, SLP04]. Darin wird gezeigt, dass sowohl Diagnosen als auch Behandlungen von Krankheiten von vielerlei Faktoren abhängen. Klassischerweise sind diese Faktoren und deren Auswirkungen auf eine Behandlung, dem behandelnden Mediziner gut bekannt. Öztürk und Aamodt zeigen jedoch auch, dass Fakten, die sie ebenfalls als *Kontext* bezeichnen und die im Zusammenhang mit bestimmten Krankheiten stehen, durch Ontologien beschrieben werden können.

Zu Beginn der 2000er Jahre wurden Ontologien dann vermehrt zur Beschreibung ubiquitärer und kontextsensitive Systeme verwendet. Belege dafür lassen sich beispielsweise in den Arbeiten von Strang [Str03], Chen [CPFJ04], Gu [GWPZ04] und Wang [WGZP04] finden. Dabei kommen alle Autoren zu dem Schluss, dass Ontologien für die Modellierung von kontextsensitiven und ubiquitären Computersystemen sehr gut geeignet sind. Insbesondere die Möglichkeit Ontologien als Wissensbasis in interdisziplinären Teams zu nutzen wird dabei geschätzt. So sagt Chen beispielsweise: „Ontologies are key requirements for building pervasive context-aware systems, in which independently developed sensors, devices, and agents are expected to share contextual knowledge and to provide relevant services and information to users based on their situational needs.“ [CFJ03]

SWRL-Regeln

Ontologien fehlte es dabei zunächst an Möglichkeiten, Zusammenhänge als einfache Regeln zu formulieren. Erst im Jahr 2004, wurde die *Semanic Web Rule Language (SWRL)* vorgestellt [HPSB+04]. Diese erlaubt es, komplexe Regeln mit Hilfe aussagenlogischer Ausdrücke unter Verwendung mathematischer Formeln, Vergleichsoperatoren und Variablen, zu formulieren. Ein Beispiel für eine solche Regel ist in SWRL-Regel 2.1 dargestellt.

SWRL-Regel 2.1: *Exemplarische Klassifikation eines Individuums der Klasse* `Uhrzeit` *in die Klasse* `Vormittag` *in Abhängigkeit der Data Property* `Stunde`

$$\texttt{a} \in \texttt{Uhrzeit} \land \texttt{a.Stunde} \geq 7 \land \texttt{a.Stunde} \leq 12 \Rightarrow \texttt{a} \in \texttt{Vormittag}$$

Annotationen

Zu jedem Element einer Ontologie können sogenannte *Annotationen* hinzugefügt werden. Sie ermöglichen es, zusätzliche Informationen an bestimmte Elemente zu knüpfen. Sie können beispielsweise genutzt werden, um einzelne Elemente, wie Klassen oder Data Properties, zu kommentieren. Bei der Auswertung der Ontologie durch einen Reasoner übernehmen die Annotationen bisher keine weiterführenden Aufgaben.

Serialisiert werden Ontologien durch die *Web Ontology Language (OWL)* [MPSP12], wie sie zum Beispiels auch von Gu [GWPZ04] und Wang [WGZP04] verwendet werden.

2.2.7 Hybride Modelle

Werden mehrere der zuvor beschriebenen Modelle miteinander kombiniert, wird von *Hybriden Modellen* gesprochen [BBH+10]. Dabei wird beabsichtigt, die Nachteile des einen Modells durch den Einsatz eines weiteren Modells auszugleichen, um kontextsensitive Systeme somit möglichst umfangreich modellieren und die Modelle zur Laufzeit möglichst effizient auswerten zu können.

In der Literatur lassen sich einige Beispiele für Hybride Modelle finden. Agostini et al. kombinieren beispielsweise Ontologien mit CC/PPs, um die Geschwindigkeit der Auswertung des Kontextmodells zu erhöhen [ABR09]. Henricksen et al. kombinieren die von ihnen entwickelte CML ebenfalls mit Ontologien mit dem Ziel, den bis dato mangelhaften Umgang von Ontologien mit unvollständigen Daten, zu verbessern [HLI04]. Auch Becker et al. argumentieren, dass die Auswertungsgeschwindigkeit von Ontologien gesteigert werden kann, wenn die in diesen hinterlegten Daten zuvor nach ihrem Ort des Auftretens verarbeitet werden (sogenannte *Spatial Models*) [BN04]. Einen ähnlichen Ansatz verfolgen auch Roussaki et al. [RSP+06]. Anstatt eines eignen Modells, kombinieren sie Ontologien jedoch mit einem Datenbank-basierten Ansatz. Auch diese beiden Ansatz können daher als Hybride Modelle betrachtet werden.

Eine gute Übersicht über Hybride Modelle und deren Vor- und Nachteile lässt sich in [BBH+10] finden.

2.3 Erkennung von Nutzerintentionen

Zur Erkennung der Intentionen des Nutzers, müssen auch diese und ihre Abhängigkeiten zum zuvor modellierten Kontext durch ein entsprechendes Modell repräsentiert werden. Um das dafür geeignetste Modell im späteren Verlauf der vorliegenden Arbeit auswählen zu können, werden in diesem Abschnitt die drei populärsten Techniken vorgestellt. Dazu zählen: Entscheidungsbäume, Künstliche Neuronale Netze (KNN) und Bayes'sche Netze.

Diese drei Techniken verfolgen auf drei verschiedenen Wegen das Ziel, unter Berücksichtigung bestimmter Eingangsgrößen eine definierte Menge an Ausgangsgrößen zu bestimmen. Im Hinblick auf die Aufgabe der Erkennung von Nutzerintentionen sind die Eingangsgrößen der aktuelle Kontext des Nutzers, während die Ausgangsgrößen die zu erkennenden Nutzerintentionen darstellen.

2.3.1 Entscheidungsbäume

Entscheidungsbäume können verwendet werden, um bestimmte Fragestellungen in Abhängigkeit zur Ausprägung gegebener Fakten zu beantworten [Bre93]. Dabei werden die Entscheidungswege in Form eines Baumes dargestellt. Knoten in diesem Baum repräsentieren dazu die für den Baum relevanten Eingangsgrößen. Die Blattknoten des Baumes repräsentieren die Ausprägungen der Ausgangsgrößen des Entscheidungsbaumes. Abbildung 2.2 veranschaulicht dieses Prinzip exemplarisch.

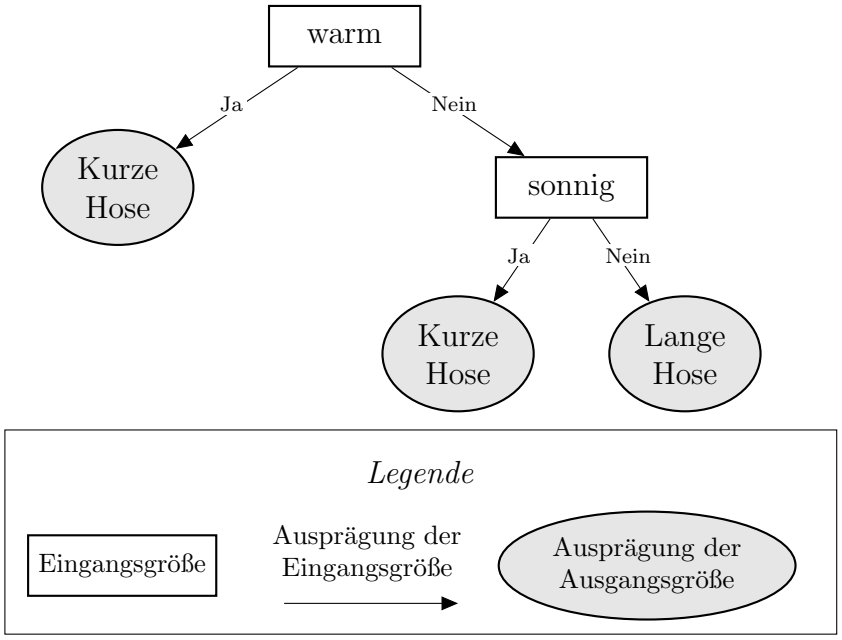

Abbildung 2.2: Exemplarischer Entscheidungsbaum zur Wahl der eigenen Bekleidung

Auf Basis der beiden Eingangsgrößen `sonnig` und `warm` soll entschieden werden, ob eine kurze oder eine lange Hose als Bekleidung gewählt wird. Die Blattknoten des Baumes tragen daher die Werte `lange Hose` oder `kurze Hose`. Der dargestellte Entscheidungsbaum zeigt, dass zunächst überprüft wird, ob die aktuelle Temperatur als warm bewertet werden kann. Falls ja, kann direkt entschieden werden, dass es sich empfiehlt eine kurze Hose zu tragen. Falls nein, wird auf der zweiten Ebene des Baumes geprüft, ob das Wetter als sonnig einzustufen ist. Ist das der Fall, wird eine kurze Hose empfohlen, andernfalls eine lange Hose. Da die Eingangsgrößen dieses Entscheidungsbaumes immer zwei mögliche Ausprägungen haben, kann dieser auch als *Binärer Entscheidungsbaum* bezeichnet werden. Es ist dadurch jedoch nicht ausgeschlossen, dass Knoten in einem Entscheidungsbaum mehr als zwei mögliche Ausprägungen besitzen.

Induktion von Entscheidungsbäumen

Zur Erstellung von Entscheidungsbäumen können diese entweder manuell oder auf Basis einer Datensammlung automatisiert erstellt werden. Die automatische Erstellung von Entscheidungsbäumen wird auch *Induktion* genannt, da von bekannten Einzelfällen auf den allgemeingültigen Entscheidungsbaum geschlossen wird.

Erste Ansätze zur Induktion wurden bereits 1964 von Sonquist und Morgan durch die sogenannten *Chi-square Automatic Interaction Detectors* beschrieben [SM64]. Ihren Namen verdanken sie der Tatsache, dass zur Bestimmung der Relevanz einer Eingangsgröße der Chi-Quadrat-Test verwendet wird. Dieser Algorithmus kann jedoch lediglich bei diskreten Eingangsgrößen, wie beispielsweise `sonnig` mit den Ausprägungen `ja` und `nein`, nicht jedoch mit kontinuierlichen Größe, wie beispielsweise der Temperatur in °C, verwendet werden. Moderne Ansätze, wie beispielsweise *ID3* [Qui86] verwenden Entropie-Berechnungen zur Bestimmung der relevanten Eingangsgrößen. Auch ID3 kann keine kontinuierlichen Eingangsgrößen verarbeiten. Sein Nachfolger *C4.5* [Qui93] unterstützt diese hingegen erstmals.

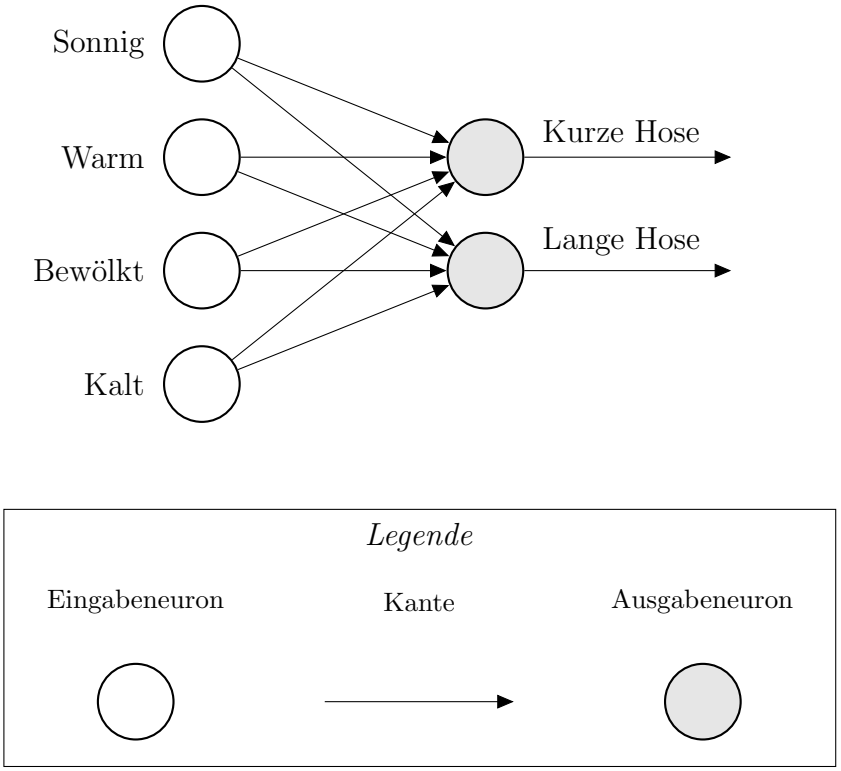

Abbildung 2.3: Exemplarisches Künstliches Neuronales Netz zur Wahl der eigenen Bekleidung

Je nach Anwendungsfall und Struktur der Eingangsgrößen stehen heutzutage daher eine Vielzahl an Algorithmen zur Induktion von Entscheidungsbäumen zur Verfügung, so dass diese nicht zwangsläufig manuell erstellt werden müssen.

Wie geeignet Entscheidungsbäume zur Modellierung von Nutzerintentionen sind, wird in Kapitel 5 überprüft.

2.3.2 Künstliche Neuronale Netze

Eine weitere Möglichkeit, die auf Basis bestimmter Eingangsgrößen auf eine definierte Menge an Ausgangsgrößen schließen kann und somit zur Vorhersage und Entscheidungsfindung genutzt werden können, sind KNNs [HMOR94]. Sie haben die aus der Biologie bekannten Neuronen zum Vorbild und funktionieren nach einem ähnlichen Grundprinzip [Roj96]. Neuronen in einem KNN sind über Kanten miteinander verknüpft. Über diese Kanten senden die einzelnen Neuronen Signale an weitere Neuronen, wenn die Summe ihrer Eingangssignale einen bestimmten Schwellwert überschreitet. Zur Gewichtung der verschiedenen Signale können die Kanten gewichtet werden.

Abbildung 2.3 zeigt exemplarisch, wie mit Hilfe eines KNN die Wahl für die eigene Bekleidung getroffen werden kann. Die möglichen Varianten aller Eingangsgrößen (`sonnig`, `warm`, `bewölkt` und `kalt`) wurden dabei als Eingabeneuronen modelliert. Die gewünschten Ausgangsgrößen (`kurze Hose` und `lange Hose`) sind als Ausgabeneuronen dargestellt worden. Aufgrund einer vollständigen Vernetzung von Eingabe- und Ausgabeneuronen könnte anschließend über die Kantengewichte und

die Schwellwerte der Ausgabeneuronen modelliert werden, welche Kanten (und damit welche Eingabeneuronen) einen Einfluss auf die jeweiligen Ausgabeneuronen haben. Es sei an dieser Stelle angemerkt, dass für dieses Beispiel auch andere KNNs möglich gewesen wären. Da die Eingangsgrößen jeweils binär sind (also nur zwei Ausprägungen haben), hätten beispielsweise die Eingangsgrößen `sonnig` und `bewölkt` auch als ein Eingabeneuron modelliert werden können, das für eine der beiden Ausprägungen feuert und für die andere nicht. Gleiches gilt für die Eingangsgrößen `warm` und `kalt`, sowie wir für die beiden Ausgangsgrößen `kurze Hose` und `lange Hose`.

In der Literatur lassen sich zahlreiche Belege dafür finden, dass KNNs zur Modellierung adaptiver Systeme eingesetzt werden. So zeigte Specht bereits 1988, dass KNNs für die Klassifikationen, Abbildungen und als assoziatives Gedächtnis verwendet werden können [Spe88]. Im gleichen Jahr zeigte Dutta et al. [DS88], dass KNNs verwendet werden können, um die Bewertung von Unternehmensanleihen vorherzusagen und dabei eine höhere Genauigkeit erzielen, als die bis dahin üblicherweise durchgeführten Regressionen. Gorr et al. [GNS94] zeigten 1994, dass KNNs in der Lage sind, die Benotung von Schülern vorherzusagen. Die dabei erzielte Genauigkeit im Vergleich zur Bewertung durch ein Komitee von Lehrern, blieb jedoch hinter den Erwartungen. Griol et al. [GC16] zeigten jüngst, dass KNNs auch eingesetzt werden können, um die Intentionen eines Nutzers während eines Sprachdialogs mit einem Computersystem zu verstehen bzw. vorherzusagen. Weitere Belege für den umfangreichen und vielseitigen Einsatz von KNNs lassen sich in den Arbeiten von Hill et al. [HMOR94] und Schmidhuber [Sch15] finden.

Trainieren Künstlicher Neuronaler Netze

Beim Trainieren eines KNN, werden mögliche Parameter so lange variiert, bis Testdatensätze aus einem jeweiligen Anwendungsfall richtig klassifiziert werden. Zu den möglichen Parametern gehören vor allem die Kantengewichte und Schwellwerte der Neuronen [Kru11]. Auf diese Art und Weise können auch KNNs, basierend auf Datensätze, so trainiert werden, dass bei bestimmten Eingangsgrößen bestimmte Ausgangsgrößen berechnet bzw. vorhergesagt werden können. Die dabei entstehenden KNNs können je ihrer grundsätzlichen Struktur für den Menschen mitunter schwer nachvollziehbar werden, da einzelnen Neuronen keine realweltlichen Fakten oder Objekte mehr zugeordnet werden können [GB10].

Weitere und detaillierte Informationen zur KNNs können entsprechenden Standardwerken, wie beispielsweise [Roj96] oder [Kru11] entnommen werden. Die Eignung von KNNs zur Modellierung von Nutzerintentionen wird ebenfalls in Kapitel 5 bewertet.

2.3.3 Bayes'sche Netze

Bayes'sche Netze [Pea85] stellen eine Möglichkeit dar, das von Bayes beschriebene Theorem [BP63] zur Berechnung bedingter Wahrscheinlichkeiten zu visualisieren. Zu diesem Zwecke wird ein gerichteter, azyklischer Graph verwendet, dessen Knoten die Variablen und dessen Kanten die bedingten Abhängigkeiten zwischen den Variablen darstellen [Kru11]. Knoten eines Bayes'schen Netzes sind zusätzlich bedingte Wahrscheinlichkeitsverteilungen zugeordnet. Anhand dieser Verteilungen kann bestimmt werden, welche Wahrscheinlichkeit für die durch diesen Knoten repräsentierte Variable, unter Berücksichtigung der durch die Kanten repräsentierten Abhängigkeiten zu

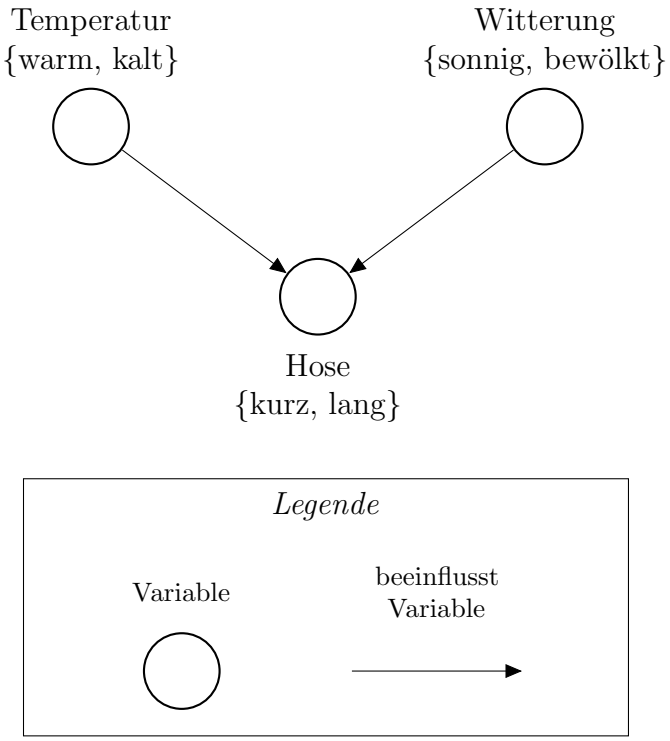

Abbildung 2.4: Exemplarisches Bayes'sches Netz zur Wahl der eigenen Bekleidung

warm	kalt
0.8	0.2

Tabelle 2.1: Exemplarische Wahrscheinlichkeitsverteilung für die Variable `Temperatur` (ergänzend zu Abbildung 2.4)

anderen Knoten bzw. Variablen, angenommen werden kann. Im einfachsten Fall sind diese Verteilungen in Form von Wahrscheinlichkeitstabellen angegeben [Kru11].

Abbildung 2.4 zeigt das bereits bekannte Beispiel als Bayes'sches Netz. Dabei ist zunächst zu erkennen, dass die Variablen `Hose`, `Temperatur` und `Witterung` als Knoten modelliert wurden. Um die Abhängigkeiten zwischen den Variablen `Temperatur` und `Hose`, sowie `Witterung` und `Hose` abzubilden, wurden Kanten zwischen den entsprechenden Knoten hinzugefügt. Die zu den Knoten gehörenden Wahrscheinlichkeitsverteilungen sind in Tabelle 2.1, Tabelle 2.2 und Tabelle 2.3 dargestellt. Da die Wahrscheinlichkeit der Variablen `Temperatur` und `Witterung` nicht von anderen Variablen abhängig ist, geben die dazugehörigen Wahrscheinlichkeitsverteilungen lediglich an, welche Wahrscheinlichkeit für den jeweiligen Wert der Variablen gilt (vgl. Tabelle 2.1 und Tabelle 2.2). So gilt beispielsweise eine Wahrscheinlichkeit von $p = 0.2$, dass die Variable `Temperatur` den Wert `kalt` annimmt. Die Wahrscheinlichkeitsverteilung für die Variable `Hose` berücksichtigt hingegen, aufgrund der modellierten Abhängigkeiten, die Werte der Variablen `Temperatur` und `Witterung` (vgl. Tabelle 2.3). So gilt beispielsweise eine Wahrscheinlichkeit von $p = 0.8$ für die Variable `Hose`, unter der Bedingung, dass die Variablen `Temperatur` und `Witterung` die Werte `warm` bzw. `bewölkt` annehmen.

sonnig	bewölkt
0.3	0.7

Tabelle 2.2: Exemplarische Wahrscheinlichkeitsverteilung für die Variable `Witterung` (ergänzend zu Abbildung 2.4)

Ausprägung `Temperatur`	Ausprägung `Witterung`	`kurz`	`lang`
warm	sonnig	0.95	0.05
warm	bewölkt	0.8	0.2
kalt	sonnig	0.6	0.4
kalt	bewölkt	0.1	0.9

Tabelle 2.3: Wahrscheinlichkeitsverteilung für die Variable `Hose` (ergänzend zu Abbildung 2.4)

Schlussfolgern in Bayes'schen Netzen

Bayes'sche Netze können je nach Anwendungsfall für verschiedenartige Schlussfolgerungen verwendet werden. Dazu zählen *kausale*, *diagnostische* und *interkausale* Schlussfolgerungen [RND10]. Bei einer kausalen Schlussfolgerung wird von einer Ursache auf einen Effekt geschlossen. Im bereits beschriebenen Beispiel könnte so beispielsweise ein konkreter Wert für die Variablen `Temperatur` und `Witterung` beobachtet werden, anhand dessen berechnet werden könnte, wie groß die Wahrscheinlichkeiten für die Werte `kurz` und `lang` der Variable `Hose` sind. Beim diagnostischen Schlussfolgern wird hingegen von einem Effekt zur Ursache geschlossen. Wird beispielsweise beobachtet, dass eine kurze Hose getragen wird und die Variable `Hose` dadurch den Wert `kurz` bekommt, würde beim diagnostischen Schlussfolgern auf die Wahrscheinlichkeiten der Werte der Variablen `Temperatur` und `Witterung` geschlossen werden. Das interkausale Schlussfolgern kann als Kombination des kausalen und diagnostischen Schlussfolgerns betrachtet werden, da von einer Ursache auf eine andere Ursache unter Berücksichtigung eines Effekts geschlossen wird. So könnte im vorliegenden Beispiel aus der Variable `Temperatur` (Ursache) auf einen Wert der Variable `Hose` (Effekt) geschlossen werden, woraufhin auf die Wahrscheinlichkeiten der Variable `Witterung` (Ursache) geschlossen werden kann. Auch beliebige Kombinationen der erwähnten Formen zur Schlussfolgerung sind möglich [RND10].

Trainieren Bayes'scher Netze

Auch Bayes'sche Netze müssen nicht notwendigerweise manuell erstellt werden, sondern können automatisiert trainiert werden. Die Struktur Bayes'scher Netze, also die Anordnung von Knoten und Kanten, kann beispielsweise mit Hilfe der Algorithmen *K2*[CH92], *CB*[SV95], *Simulated Annealing*[Bou95], oder *TAN*[FGG97] automatisiert gelernt werden. Zum Lernen der Wahrscheinlichkeitsverteilungen der

Knoten eines Bayes'sche Netzes können anschließend Algorithmen wie Bayesian Model Averaging (BMA)[9] eingesetzt werden.

In der Literatur lassen sich bereits Arbeiten finden, die Bayes'sche Nete zur Modellierung in der Automobilbranche einsetzen. So untersuchte Ablaßmeier [APRR07, Abl09], wie Bayes'sche Netze in einem Agenten-basierten Ansatz verwendet werden können, um ein multimodales, kontextadaptives Informationsmanagementsystem umzusetzen. Im Unterschied zur vorliegenden Arbeit fokussiert er dabei auf einzelne Anwendungsfälle und nicht auf den Modellierungs- und Entwicklungsprozess. Schroven [Sch11] zeigt, das Bayes'sche Netze verwendet werden können, um Fahrmanöver des Fahrers vorherzusagen.

Auch die Eignung Bayes'scher Netze zur Modellierung von Nutzerintentionen wird in Kapitel 5 näher untersucht und mit Entscheidungsbäumen und KNNs verglichen.

2.4 Zusammenfassung

In diesem Kapitel wurde grundlegendes, für das Verständnis der weiteren Kapitel dieser Arbeit erforderliches, Wissen vermittelt. Dabei wurde in Abschnitt 2.1 zunächst erläutert, in welche Kategorien sich die Daten in Fahrzeugen klassifizieren lassen und wie diese kommuniziert und verarbeitet werden können. Auch das Kommunikationsprotokoll EXLAP und die OSGi Service Plattform wurden im gleichen Abschnitt vorgestellt. Anschließend wurde in Abschnitt 2.2 diskutiert, welche Möglichkeiten zur Modellierung kontextsensitiver Systeme sich in der Literatur finden lassen. Abschließend wurden in Abschnitt 2.3 verschiedene Techniken zur Modellierung von Nutzerintentionen vorgestellt.

[9] Laut Eicher et al. [EPR11] lässt sich das BMA nicht auf einen Autor zurückführen, sondern geht auf Leamer [Lea78], Raftery [Raf88], Madigan und Raftery [MR94], Raftery[Raf93], sowie George und McCulloch [GM93] zurück. Eine umfassende Zusammenfassung zum BMA lässt sich außerdem in der Arbeit von Hoeting et al. finden [HMRV99].

3. Anforderungsanalyse

„It is no use saying, 'We are doing our best.'
You have got to succeed in doing what is necessary.“

(Winston S. Churchill [CJ80])

In diesem Kapitel werden Anforderungen ermittelt und analysiert, die ein kontext- und intentionssensitives FIS und dessen Modellierung erfüllen muss. In Abschnitt 3.1 werden dazu zunächst grundlegende Begriffe wie *Stakeholder*, *Funktionale Anforderung*, *Qualitätsanforderung* und *Randbedingung* definiert. Anschließend werden in Abschnitt 3.2 Stakeholder ermittelt, die Anforderungen an ein System zur Erkennung von Kontextinformationen und Nutzerintentionen stellen. Auf Basis dieser Stakeholder wurde, gemeinsam mit Domänenexperten auf dem Gebiet der Entwicklung von FIS, eine Anforderungsanalyse anhand von Experteninterviews durchgeführt. Die Ergebnisse dieser Analyse werden in Abschnitt 3.3 vorgestellt. Abschnitt 3.4 fasst das Kapitel abschließend zusammen.

3.1 Begriffsdefinitionen

Bevor konkrete Anforderungen erhoben werden, die ein kontext- und intentionssensitives FIS und deren Modellierung erfüllen müssen, werden in diesem Abschnitt einige grundlegende Begrifflichkeiten definiert. Zunächst soll geklärt werden, was genau unter dem Begriff *Anforderung* im Allgemeinen verstanden wird. Der International Standard 24765 des Institute of Electrical and Electronics Engineers (IEEE) definiert eine Anforderung wie folgt:

Definition 3.1: *Anforderung*

Eine Anforderung ist:

(1) Eine Bedingung oder Fähigkeit, die von einem Nutzer zur Erreichung eines Ziels benötigt wird.

(2) Eine Bedingung oder Fähigkeit, die von einem System, einer Systemkomponente, einem Produkt oder einem Dienst erfüllt werden muss, um einer Vereinbarung, einer Norm, einer Spezifikation oder anderen formalen Dokumenten zu genügen.

(3) Eine dokumentierte Darstellung einer Bedingung oder Fähigkeit wie in (1) oder (2) beschrieben.

(4) Eine Bedingung oder Fähigkeit, die von einem System, einer Systemkomponente, einem Produkt, einem Dienst oder einem Ergebnis erfüllt werden muss, um einer Vereinbarung, einer Norm, einer Spezifikation oder anderen formalen Dokumenten zu genügen. Anforderungen enthalten die quantifizierten und dokumentierten Notwendigkeiten, Bedürfnisse und Erwartungen des Auftraggebers, des Kunden und anderer Interessengruppen (*Stakeholder*).

(Übersetzt aus IEEE Std 24765-2010 [IEE10])

Diese Definition macht deutlich, dass Anforderungen gut dokumentierte Eigenschaften eines Systems sind, die zur Erreichung bestimmter Ziele dienen. Punkt (4) eben dieser Definition hebt hervor, dass Anforderungen quantifizierbar sein sollten.

Im gleichen Standard [IEE10] definiert das IEEE auch den Begriff der bereits erwähnten *Stakeholder* (zu Deutsch *Interessengruppe*), wie folgt:

Definition 3.2: *Stakeholder*

Ein Stakeholder ist:

(1) Ein Individuum oder eine Organisation mit einem Recht, einem Anteil, einem Anspruch oder einem Interesse an einem System oder dessen Besitz, dass ihre Bedürfnisse und Erwartungen erfüllt.

(2) Ein Individuum, eine Gruppe oder eine Organisation, die von Risiken betroffen sein kann, oder diese selbst beeinflussen kann.

(3) Ein Individuum, eine Gruppe oder eine Organisation, die von Entscheidungen und Aktivitäten betroffen sein kann, oder diese selbst beeinflussen kann.

(4) Eine Person oder eine Organisation (zum Beispiel Kunden, Auftraggeber, ausübende Organisationen oder die Öffentlichkeit), die aktiv in ein Projekt eingebunden sind, oder deren Interesse positiv als auch negativ von der Ausführung des Projektes betroffen sein kann. Stakeholder können auch Einfluss auf das Projekt und dessen Ergebnisse ausüben.

(Übersetzt aus IEEE Std 24765-2010 [IEE10])

Stakeholder eines Systems sind also Personen, Gruppen oder Organisationen, die Berührungspunkte mit dem System haben und Einfluss auf die Anforderungen haben können. Rupp definiert eine *exzellente* Anforderung [Rup01] ferner als:

- vollständig,
- korrekt,
- klassifizierbar bezüglich der juristischen Verbindlichkeit,
- konsistent gegenüber anderen Anforderungen und in sich,
- testbar,
- verständlich für alle Stakeholder,
- umsetzbar und realisierbar,
- notwendig,
- verfolgbar,
- bewertbar und
- eindeutig.

Bei der Erhebung von Anforderungen ist es daher von erhöhter Wichtigkeit, Anforderungen klar und unmissverständlich zu formulieren, den jeweiligen Stakeholder zu benennen und ein quantifizierbares Erfüllungskriterium zu definieren.

In der Literatur [Yeh82, Par98, PR11] werden Anforderungen ferner in verschiedene Arten unterteilt: *funktionale Anforderungen*, *nicht funktionale Anforderungen* (auch *Qualitätsanforderungen* [PR11] oder *constraints* [Rom85] genannt) und *Randbedingungen*. Die erste dieser drei Arten – die funktionalen Anforderungen – definieren, welche Struktur, welche Funktion und welches Verhalten für das zu entwickelnde System gelten sollen. Pohl und Rupp definieren funktionale Anforderungen daher wie folgt:

Definition 3.3: *Funktionale Anforderung*

Eine funktionale Anforderung ist eine Anforderung bezüglich des Ergebnisses eines Verhaltens, das von einer Funktion des Systems bereitgestellt werden soll.

(Nach [PR11])

Qualitätsanforderungen hingegen definieren, in welcher Qualität die funktionalen Anforderungen vom System erfüllt werden sollen und bedingen daher den funktionalen Anforderungen [Par98]. Sie haben mitunter größere Auswirkungen auf die Systemarchitektur, als die funktionalen Anforderungen und beinhalten typischerweise Anforderungen zur Performance, Verfügbarkeit, Zuverlässigkeit, Skalierbarkeit und Portierbarkeit des Systems [PR11]. Pohl und Rupp definieren Qualitätsanforderungen wie folgt:

Definition 3.4: *Qualitätsanforderung*

Eine Qualitätsanforderung ist eine Anforderung, die sich auf ein Qualitätsmerkmal bezieht, das nicht durch funktionale Anforderungen abgedeckt wird.

(Nach [PR11])

Zusätzlich zu funktionalen Anforderungen und Qualitätsanforderungen, können Randbedingungen gewissermaßen den Rahmen der Umsetzung eines Systems vor geben. Hierzu können zum Beispiel die Verwendung bestimmter Programmiersprachen zur Implementierung oder ein bestimmtes Fertigstellungsdatum zählen [PR11].

Definition 3.5: *Randbedingungen*

Eine Randbedingung ist eine Anforderung, die den Lösungsraum jenseits dessen einschränkt, was notwendig ist, um die funktionalen Anforderungen und die Qualitätsanforderungen zu erfüllen.

(Nach [PR11])

Um die im Rahmen dieser Dissertation ermittelten Anforderungen nach obigem Schema einordnen zu können, werden im nächsten Schritt zunächst alle Stakeholder eines kontext- und intentionssensitivem FIS ermittelt.

3.2 Stakeholder

Unabhängig vom System oder vom Projekt ist es von enormer Wichtigkeit alle Stakeholder zu erfassen, andernfalls kann nicht gewährleistet werden, dass auch alle Anforderungen erfasst werden. Dies führt im schlimmsten Fall zu Änderungsanträgen nach der Inbetriebnahme eines Systems, welche hohe Integrationskosten nach sich ziehen können [PR11]. Um alle Stakeholder zu ermitteln, schlägt Rupp eine *Checkliste von Stakeholderklassen*[1] abzuarbeiten [Rup01].

In diesem Abschnitt werden basierend auf eben dieser Checkliste relevante Stakeholder vorgestellt, die gemäß der in Abschnitt 3.1 präsentierten Definitionen, direkt oder indirekt Anforderungen an ein kontext- und intentionssensitives FIS stellen. Tabelle 3.1 fasst die ermittelten Stakeholder und deren Beschreibung zusammen.

Tabelle 3.1: Übersicht der ermittelten Stakeholder eines kontext- und intentionssensitivem Fahrzeuginformationssystem

Name des Stakeholders	Beschreibung
Management	Das Management stellt sicher, dass die Umsetzung des kontext- und intentionssensitivem FIS und dessen Eindruck vor dem Anwender mit den Zielen und der Strategie der Volkswagen AG übereinstimmt und entscheidet über eine etwaige Integration in das finale Produkt Fahrzeug.

[1] Auch online abrufbar unter: http://tinyurl.com/checkstake. Zuletzt überprüft am 19.10.2016.

Name des Stakeholders	Beschreibung
Anwender	Der Anwender nutzt das kontext- und intentionssensitive FIS während er seiner primären Aufgabe – dem Führen eines Fahrzeugs – nachgeht. Er profitiert direkt von allen Vorteilen des finalen Systems und steht je nach Nutzung mitunter in täglichem Kontakt zum System.
Käufer	Der Käufer trifft die Kaufentscheidung für das Fahrzeug und somit auch für das kontext- und intentionssensitive FIS. Der Käufer ist nicht notwendigerweise mit dem Anwender gleichzusetzen. Insbesondere bei Geschäftsfahrzeugen kann der Käufer ein Einkäufer sein, der für mehrere Anwender mehrere Fahrzeuge bestellt bzw. kauft.
Servicemitarbeiter	Der Servicemitarbeiter setzt sich nach dem Verkauf des Produkts mit eventuellen Beanstandungen durch den Anwender oder den Käufer auseinander.
Verkaufs-mitarbeiter	Der Verkaufsmitarbeiter präsentiert und verkauft das Produkt an den Käufer. Zusätzlich berät er den Kunden falls gewünscht bei der Konfiguration des Fahrzeugs.
Vertrieb & Marketing der Volkswagen AG	Der Vertrieb und das Marketing der Volkswagen AG bewerben und vertreiben das kontext- und intentionssensitve FIS. Ferner sind ihnen Kunden- und Marktbedürfnisse aus Meinungsumfragen und Studien bekannt.
System-Entwickler	Der System-Entwickler hat die Aufgabe das kontext- und intentionssensitive FIS zu entwickeln. Er definiert Schnittstellen zum System und kann Schnittstellen anderer Systeme durch selbst aufgestellte Anforderungen beeinflussen.
System-Integrator	Der System-Integrator muss sicherstellen, dass sich das kontext- und intentionssensitive FIS in das Gesamtsystem Fahrzeug integrieren lässt.
Bedienkonzept-Entwickler	Der Bedienkonzept-Entwickler konzipiert das Bedienerlebnis des kontext- und intentionssensitiven FIS. Er legt fest, welche Anwendungsfälle durch das System abgedeckt werden sollen und wie Anzeige und Bedienung des Systems (Human-Machine Interface (HMI)) der einzelnen Anwendungsfälle erfolgen.

Die vorgestellte Liste der Stakeholder erhebt an dieser Stelle keinen Anspruch auf generelle Vollständigkeit, da die in der vorliegenden Dissertation erarbeiteten Konzepte und Systeme im Bereich der Forschung und nicht dem der Serienentwicklung eines kontext- und intentionssensitiven FIS entstanden sind. Bei der Überführung der Konzepte und Systeme in ein Serienprodukt empfiehlt sich daher eine erweiternde Analyse der Stakeholder. Weitere potentielle Stakeholder könnten zum Beispiel sein: *der Gesetzgeber*, der mitunter allgemeine Anforderungen an ein FIS erhebt (zum Beispiel zur Einschränkung der Ablenkung von der Fahraufgabe), *die öffentliche Meinung* zur kontext- und intentionssensitiven Systemen, die negativen aber auch positiven Druck auf die Einführung eines entsprechenden Systems aufbringen kann, oder *Standards und Standardgremien*, die zum Beispiel interessiert daran sein könnten eine möglichst hohe Interoperabilität zwischen verschiedenen Herstellern zu gewährleisten.

Nichtsdestotrotz umfasst die präsentierte Liste alle, für die prototypische Entwicklung eines kontext- und intentionssensitiven FIS und dazugehörigen Modellierungstechniken, relevanten Stakeholder, die direkt oder indirekt Anforderungen verschiedener Art erheben.

3.3 Ermittlung von Anforderungen

In diesem Abschnitt werden Anforderungen präsentiert, die von den in Abschnitt 3.2 vorgestellten Stakeholdern an ein kontext- und intentionssensitives FIS erhoben werden.

Die Anforderungen werden dabei zunächst wie folgt gegliedert: in Abschnitt 3.3.1 werden Funktionale Anforderungen an die Verarbeitung von Kontextinformationen in einem kontext- und intentionssensitiven FIS vorgestellt. In Abschnitt 3.3.2 werden dann Anforderungen an die Verarbeitung und Erkennung von Nutzerintentionen präsentiert. Abschließend werden in Abschnitt 3.3.3 Qualitätsanforderungen an die Modelle von Kontextinformationen und Nutzerintentionen vorgestellt.

3.3.1 Verarbeitung von Kontextinformationen

A01: Verknüpfung mit bereits vorhanden Daten

Stakeholder:	Bedienkonzept-Entwickler, System-Integrator
Klassifikation:	Funktionale Anforderung

Ein System zur Verarbeitung von Kontextinformationen bezieht Daten aus verschieden Quellen. Die wohl grundlegendste Anforderung an ein solches System ist daher wohl, dass es sich einfach (d.h. ohne neue, zusätzliche Steuergeräte und ohne Verbindungen zwischen solchen) mit im Fahrzeug bereits vorhanden Steuergeräten und damit auch mit deren Daten verknüpfen lässt.
Ein System zur Verarbeitung von Kontextinformationen sollte Zugang zu sämtlichen Daten im Fahrzeug haben, die anderen Systemen im Fahrzeug bereits zur Verfügung stehen.

A02: Abstraktion & Aggregation

Stakeholder:	Bedienkonzept-Entwickler
Klassifikation:	Funktionale Anforderung

Einzelne Kontextinformationen (in der Regel Messwerte eines Sensors) sollen im Rahmen der Verarbeitung von Kontextinformationen zu höherwertigen Kontextinformationen bis hin zu einer komplexen Situation aggregiert und abstrahiert werden können. Wie diese Abstraktion bzw. Aggregation von Kontextinformationen im Detail stattfindet, soll vom *Bedienkonzept-Entwickler* für jeden Anwendungsfall individuell definiert werden können.

A03: Abbilden von Regeln

Stakeholder:	Bedienkonzept-Entwickler
Klassifikation:	Funktionale Anforderung

Der Bedienkonzept-Entwickler soll die Möglichkeit haben, mit Hilfe von Regeln festlegen zu können, wie Kontextinformationen zur Laufzeit vom System verarbeitet werden. So soll er zum Beispiel die Möglichkeit bekommen, durch Angabe bestimmter Vorbedingungen festzulegen, zu welcher höherwertigen Kontextinformation bzw. Situation einfache Kontextinformationen abstrahiert werden. Die Anforderung, dies regelbasiert durchzuführen, liegt darin begründet, dass die Denkmuster der Bedienkonzept-Entwickler nach eben solchen Regeln aufgebaut sind.

A04: Unterstützung verschiedener Datentypen

Stakeholder:	System-Entwickler, System-Integrator
Klassifikation:	Qualitätsanforderung

Kontextinformationen können sich in ihrem Datentyp unterscheiden. Möglich sind primitive Datentypen wie Bool'sche Kontextinformationen (wie zum Beispiel der Status eines angelegten bzw. nicht-angelegten Sicherheitsgurtes), Ganzen Zahlen (wie zum Beispiel die Anzahl belegter Sitze), Rationalen Zahlen (wie zum Beispiel die Außentemperatur), bis hin zu komplexen Datentypen, die sich aus einer Menge dieser primitiven Datentypen zusammensetzen.
Ein System zur Verarbeitung von Kontextinformationen sollte daher die folgenden Datentypen unterstützen: *Boolean*, *Integer*, *Double*, *String*, sowie komplexe Datentypen als Kombination dieser primitiven Datentypen.

A05: Unterstützung verschiedener Änderungsfrequenzen

Stakeholder:	System-Entwickler, System-Integrator
Klassifikation:	Qualitätsanforderung

Kontextinformationen können sich auch in der Frequenz ihrer Änderung unterschieden. So existieren Kontextinformationen, die sich über die gesamte Lebensdauer des Fahrzeugs hinweg nicht ändern (wie zum Beispiel die Fahrzeugklasse oder die Anzahl

der Türen), aber auch Kontextinformationen die sich mehrmals in der Sekunde ändern können (wie zum Beispiel die Fahrzeuggeschwindigkeit oder die Motordrehzahl).
Ein System zur Verarbeitung von Kontextinformationen sollte daher unabhängig von der Frequenz der Änderung der Kontextinformationen operieren.

A06: Klassifikationen von Kontextinformationen

Stakeholder:	Bedienkonzept-Entwickler
Klassifikation:	Funktionale Anforderung

Kontextinformationen sollen durch eine individuell festgelegte Klassifikation in bestimmte Klassen eingeteilt werden können. Die vom Regensensor gemessene Menge an Wasser auf der Windschutzscheibe könnte so beispielsweise in *wenig*, *mittel* oder *viel* klassifiziert werden. Eine solche Klassifikation soll die Weiterverarbeitung der aggregierten Kontextinformationen vereinfachen, da anstatt mit Sensorrohdaten mit deren Klassifikation gearbeitet werden kann.

A07: Kontextinformationen aus multiplen Quellen

Stakeholder:	System-Entwickler, System-Integrator
Klassifikation:	Funktionale Anforderung

Kontextinformationen können aus verschiedenen Quellen stammen. Auch ein und die selbe Kontextinformation kann aus verschiedenen Quellen stammen. So kann die Geschwindigkeit des Fahrzeugs beispielsweise klassisch per Drehgeber oder aber auch mittels GPS gemessen werden.
Ein System zur Verarbeitung von Kontextinformationen sollte daher die Möglichkeit bieten, mehre Quellen von Kontextinformationen anzubinden und ein und die selbe Kontextinformation aus verschiedenen Quellen beziehen zu können.

A08: Kontextinformationen aus unsicheren Quellen

Stakeholder:	System-Entwickler, System-Integrator
Klassifikation:	Funktionale Anforderung

Eine Kontextinformation kann, je nachdem aus welcher Quelle sie stammt, mit einer gewissen Unsicherheit belastet sein. Die Ermittlung der Fahrzeuggeschwindigkeit mittels Drehgeber beispielsweise ist in der Regel deutlich genauer und somit sicherer als die Ermittlung der Geschwindigkeit mittels GPS.
Ein System zur Verarbeitung von Kontextinformationen sollte daher in der Lage sein, Unsicherheiten von Kontextinformationen individuell für jede Quelle von Kontextinformationen zu berücksichtigen.

A09: Verarbeitung von Kontextinformationen aus unsicheren Quellen

Stakeholder:	System-Entwickler, System-Integrator
Klassifikation:	Funktionale Anforderung

Werden mit Unsicherheiten belastete Kontextinformationen zur Abstraktion oder Aggregation von neuen Kontextinformationen verwendet, so sollen deren Unsicherheiten Einfluss auf die Unsicherheit der neuen Kontextinformation haben. Dabei soll gelten: je größer die Unsicherheit der ursprünglichen Kontextinformationen, desto größer die Unsicherheit der aggregierten Kontextinformation.

A10: Partielle Auswertung

Stakeholder:	System-Entwickler
Klassifikation:	Funktionale Anforderung

Um die Geschwindigkeit der Auswertung möglichst hoch zu halten, fordert der System-Entwickler, dass die Auswertung des Kontextmodells bei veränderter Datenlage der einzelnen Kontextinformationen auch partiell durchgeführt werden sollen können. Hintergrund dieser Anforderung ist, dass unter Berücksichtigung aller Sensoren aus den Bereichen Fahrer, Fahrzeug und Umwelt, Modelle mit sehr vielen Kontextinformationen erwartet werden, die jedoch keinesfalls alle abhängig von einander sind. Unabhängige Kontextinformationen sollten daher getrennt voneinander ausgewertet werden können.

A11: Unvollständige Daten

Stakeholder:	System-Integrator
Klassifikation:	Funktionale Anforderung

Mit der Modellierung der Kontextinformationen wird das Ziel verfolgt, die reale Welt möglichst umfangreich zu erfassen und zur Laufzeit des Systems bewerten zu können. Dazu werden mitunter Kontextinformationen und Situationen modelliert, die zur Laufzeit nicht zu jeder Zeit vorhanden oder verfügbar sind. Das System zur Verarbeitung von Kontextinformationen muss daher in der Lage sein, mit einer unvollständigen Datenlage umgehen zu können. Dieser Umgang soll sich insbesondere dadurch auszeichnen, dass modelliertes Verhalten, welches nicht von eben diesen fehlenden Daten abhängig ist, dennoch fehlerfrei zur Laufzeit umgesetzt wird.

A12: Modell als Austausch-Medium

Stakeholder:	System-Entwickler
Klassifikation:	Qualitätsanforderung

Die bei der Entwicklung kontextsensitiver FIS beteiligten Mitarbeiter, vefügen in der Regel über einen voneinander abweichenden Kenntnisstand zu bestimmten Vorgängen innerhalb des Systems. Während ein Bedienkonzept-Entwickler sehr gute Kenntnisse darüber besitzt, welche Erwartungen spätere Nutzer des kontextsensitiven Systems

haben, so besitzt der System-Integrator beispielsweise sehr gute Kenntnisse darüber, welche Hard- und Software-Komponenten benötigt werden, um eben diese Erwartungen der Nutzer zu erfüllen. Bei der Entwicklung komplexer Systeme ist es von erhöhter Bedeutung, diese Interdisziplinarität zwischen den Mitarbeitern durch ein zentrales Zielbild zu unterstützen. In Bezug auf die Kontextinformationen, soll das Modell diese Aufgabe übernehmen können. Das bedeutet, dass es als Austausch-Medium zwischen den verschiedenen beteiligten Parteien dient und dabei als verbindliches *Dokument* gilt, dass das zu entwickelnde System zentral und umfangreich beschreibt.

A13: Erweiterbarkeit

Stakeholder:	Management
Klassifikation:	Qualitätsanforderung

Das Design und die Entwicklung eines Systems zur Verarbeitung von Kontextinformationen soll zukunftssicher sein. So soll es beispielsweise möglich sein neue Kontextinformationen in das bestehende System zu integrieren, ohne die Implementierung des Systems zu verändern. Auch das Hinzufügen neuer Quellen von Kontextinformationen oder deren Austausch soll ohne eine Änderung der Implementierung möglich sein.

3.3.2 Erkennung von Nutzerintentionen

A14: Wahrscheinlichkeit der Nutzerintention

Stakeholder:	Bedienkonzept-Entwickler
Klassifikation:	Funktionale Anforderung

Die Erkennung von Nutzerintentionen wird durch eine Vielzahl von Faktoren beeinflusst. Werden nicht alle Faktoren bei der Erkennung berücksichtigt (zum Beispiel weil nicht alle Faktoren bekannt sind, die einen Einfluss auf eine mögliche Nutzerintention haben), kann die Erkennung einer Nutzerintention nur mit einer bestimmten Wahrscheinlichkeit korrekt sein.
Ein System zur Erkennung von Nutzerintentionen muss daher angeben können, mit welcher Wahrscheinlichkeit eine vorhergesagte Nutzerintentionen zutreffend ist.

A15: Mehrere Nutzerintentionen zu einem Zeitpunkt

Stakeholder:	Bedienkonzept-Entwickler
Klassifikation:	Funktionale Anforderung

Nutzer können mehr als eine Intention zur gleichen Zeit haben. So kann ein Nutzer zum Beispiel die Intention haben, zu einem bestimmten Ziel zu fahren und gleichzeitig die Intention haben, einen bestimmten Kontakt anzurufen.
Ein System zur Erkennung von Nutzerintentionen muss daher in der Lage sein, mehrere Nutzerintentionen parallel zu erkennen.

A16: Nutzerintentionen sind situationsabhängig

Stakeholder: Bedienkonzept-Entwickler

Klassifikation: Funktionale Anforderung

Der Bedienkonzept-Entwickler fordert, dass Nutzerintentionen, in Abhängigkeit der durch das System zur Verarbeitung von Kontextinformationen erkannten Situationen, modelliert werden können müssen. Das bedeutet, dass ihm während der Modellierung der Nutzerintentionen die Möglichkeit eingeräumt werden muss, auf die modellierten Kontextinformationen und Situationen zurückgreifen zu können und, dass das System zur Erkennung von Nutzerintentionen mit dem System zur Verarbeitung von Kontextinformationen zur Laufzeit in Verbindung steht.

A17: Individuelle Anpassung an Nutzer

Stakeholder: Bedienkonzept-Entwickler

Klassifikation: Funktionale Anforderung

Nutzerintentionen variieren je nach Nutzer und dessen Gewohnheiten.
Ein System zur Erkennung von Nutzerintentionen soll sich individuell an das Verhalten seines jeweiligen Nutzers anpassen. Die Erkennung von Nutzerintentionen soll auf zuvor beobachteten Nutzerverhalten beruhen.

A18: Vergessen von gelerntem Nutzerverhalten

Stakeholder: Bedienkonzept-Entwickler

Klassifikation: Funktionale Anforderung

Ändert ein Nutzer sein Verhalten in bestimmten Situationen und hat er somit in gleichen Situationen andere Intentionen, so soll das System zur Erkennung von Nutzerintentionen in der Lage sein, gelerntes Wissen über seinen Nutzer wieder *zu vergessen*. Aktuelleres Nutzerverhalten soll eine höhere Priorität bei der Erkennung von Nutzerintentionen bekommen, als länger zurückliegendes Nutzerverhalten.

3.3.3 Modellierung

A19: Verständlichkeit

Stakeholder: Bedienkonzept-Entwickler

Klassifikation: Qualitätsanforderung

Die Entwicklung eines Systems zur Verarbeitung von Kontextinformationen soll auch von Personen durchgeführt werden können, die typischerweise keine Software-Entwickler oder Programmierer sind (zum Beispiel Bedienkonzept-Entwickler).

A20: Standardisierung

Stakeholder: Management

Klassifikation: Qualitätsanforderung

Um sowohl den initialen, als auch den Pflegeaufwand zu minimieren, sollen zur Modellierung von kontext- und intentionssensitiven FIS (soweit wie möglich) standardisierte Lösungen eingesetzt werden.

A21: Tool-Unterstützung

Stakeholder:	Bedienkonzept-Entwickler, System-Entwickler
Klassifikation:	Qualitätsanforderung

Das Modellieren von kontext- und intentionssensitiven FIS soll nach Möglichkeit durch bereits bestehende Tools erfolgen können. Dabei soll das Tool den System-Entwickler bei alltäglich wiederkehrenden Aufgaben durch Teilautomatisierungen unterstützen und einem Bedienkonzept-Entwickler die Möglichkeit eröffnen am Entwicklungsprozess auch ohne tiefgehende technische Kenntnisse teilnehmen zu können.

A22: Aktualität

Stakeholder:	System-Entwickler
Klassifikation:	Qualitätsanforderung

Die verwendeten Modellen sollen einer gewissen Aktualität unterliegen, damit davon ausgegangen werden kann, dass diese aktiv weiterentwickelt werden und zukünftige neue Trends und wissenschaftliche Erkenntnisse in diese integriert werden.

3.4 Zusammenfassung

In diesem Kapitel wurden Stakeholder und Anforderungen eines kontext- und intentionssensitivem FIS ermittelt. Die ermittelten Anforderungen dienen im weiteren Verlauf dieser Arbeit als Grundlage für Kapitel 4 und Kapitel 5, um Techniken und Methoden zur Modellierung von Kontextinformationen und Nutzerintentionen zu entwickeln. In Kapitel 7 werden die in diesem Kapitel ermittelten Anforderungen auf ihre Erfüllung hin überprüft werden.

4. Modellierung von Kontextinformationen

„For me CONTEXT is the key – from that comes the understanding of everything.“

(Kenneth Noland, US-amerikanischer Künstler, [Nol88])

Kontextinformationen sind Daten und Fakten, die dazu beitragen, die aktuelle Situation in der sich ein Nutzer befindet, näher zu beschreiben. Die Entwicklung kontextsensitiver FIS ist daher darauf angewiesen, ein umfassendes Verständnis dieser Daten zu besitzen, um das Systemverhalten, der aktuellen Situation des Nutzers entsprechend, anpassen zu können. Die Tatsache, dass sowohl die Situationen, als auch die ihnen zugrunde gelegten Kontextinformationen in ihrer Syntax und Semantik hochgradig vielfältig sein können, steht in Konkurrenz zu der Anforderung, kontextsensitive Systeme möglichst schnell und effizient neuen Bedienkonzepten anzupassen. Dieses Kapitel zeigt, wie dieser Konflikt mit Techniken und Methoden der modellbasierten Software-Entwicklung gelöst werden kann.

In einem ersten Schritt werden dazu in Abschnitt 4.1 verschiedene, bereits bestehende, Ansätze zur Modellierung von Kontextinformationen untersucht und verglichen. Dabei wird herausgearbeitet, dass Ontologiebasierte Modelle nach heutigem Stand der Technik eine gute Grundlage für die Modellierung kontextsensitiver Systeme bilden. In Abschnitt 4.2 wird anschließend erläutert, wie Ontologien genutzt werden können, um Kontextinformationen für kontextsensitive FIS zu modellieren. Dabei wird detailliert auf die Modellierung von Kontextinformationen, Situationen, sowie Beziehungen zwischen diesen eingegangen. Im gleichen Abschnitt wird auch dargelegt, welche Erweiterungen im Rahmen dieser Arbeit durchgeführt wurden, um die in Kapitel 3 gestellten Anforderungen an die Modellierung von Kontextinformationen zu erfüllen. Abschnitt 4.3 fasst das Kapitel abschließend zusammen.

Die in diesem Kapitel vorgestellten Arbeiten wurden bereits in [LBS14] und [LSSS16] veröffentlicht.

4.1 Auswahl einer geeigneten Modellierung

In diesem Abschnitt wird die nach heutigem Stand der Technik erfolgversprechendste Methode zur Modellierung von Kontextinformationen ausgewählt. Zu diesem Zweck werden Arbeiten aus der Literatur herangezogen, die verschiedenen Techniken zur Modellierung von Kontextinformationen verglichen haben. Dabei werden insbesondere solche Arbeiten näher betrachtet, deren Vergleichskriterien eine große Überschneidung mit den im Rahmen dieser Arbeit ermittelten Anforderungen aufweisen, um annehmen zu können, dass die in diesen Vergleichen positiv bewerteten Techniken auch für die Modellierung von Kontextinformationen für FIS gut geeignet sind, beziehungsweise negativ bewerte Techniken nicht geeignet sind. Dazu werden die angesetzten Vergleichskriterien in Relation zu den in Kapitel 3 ermittelten Anforderungen gesetzt. Die für diesen Abschnitt relevanten Arbeiten setzen sich zusammen aus einem Vergleich aus dem Jahre 2004 von Strang et al. [SLP04], einer 2010 erstellten, umfassenden Übersicht verschiedener Techniken von Bettini et al. [BBH+10], sowie einem ergänzender Vergleich von Bergmann aus dem Jahr 2014 [Ber14].

Da die Techniken und Methoden, die in diesem Abschnitt verglichen werden, bereits detailliert in Abschnitt 2.2 vorgestellt wurden, wird auf eine erneute Beschreibung an dieser Stelle verzichtet.

4.1.1 Vergleich von Strang et al.

Ein erster Vergleich verschiedener Techniken zur Modellierung von Kontextinformationen wurde von Strang et al. im Jahre 2004 durchgeführt [SLP04]. Dabei wurden zunächst die folgenden Vergleichskriterien definiert:

- **Verteilter Aufbau**
  Strang et al. argumentieren, dass kontextsensitive Systeme zur Klasse ubiquitärer Computersysteme gehören und damit als Nachfolger mobiler Computersysteme einzustufen sind. Mobilen Computersystemen wiederum vorangegangen, sind verteilte Computersysteme, bei denen die Systemleistung keine zentrale Recheneinheit übernimmt, sondern eine Vielzahl verschiedener Komponenten auf unterschiedlicher Hardware. Folglich muss daher davon ausgegangen werden, dass auch kontextsensitiven Systemen ein gewisser Grad an *Verteiltheit* zugrunde liegt und sie keine zentrale Instanz haben, sondern der Aufbau und die Administration von Daten verteilt geschieht. [SLP04]

  Diese von Strang et al. formulierte Anforderung findet sich auch in den im Rahmen dieser Arbeit ermittelten Anforderungen wieder. So beschreibt Anforderung A07, dass Kontextinformationen aus verschiedenen Quellen stammen, weshalb eine verteilter Aufbau notwendig wird. Auch die Möglichkeit zur modularen Erweiterung (vgl. Anforderung A13) erfordert einen verteilten Aufbau eines kontextsensitiven Systems, sowie einer verteilten Art der Modellierung von Kontextinformationen.

- **Partielle Auswertung**
  Aufgrund der Vielfalt und Menge der Daten, die ein kontextsensitives System verarbeiten muss, ist es laut Strang et al. erforderlich, dass eine Technik

und Methodik zur Modellierung kontextsensitiver Systeme es erlaubt, zur Laufzeit lediglich einige wenige Teile, anstatt zu jeder Zeit das Gesamtmodell auszuwerten. [SLP04]

Der Bedarf einer partiellen Auswertung wurde auch im Rahmen der Anforderungsanalyse dieser Arbeit ermittelt (vgl. Anforderung A10). Auch wenn die Möglichkeit zur partiellen Auswertung unter Umständen erst zur Laufzeit eines kontextsensitiven Systems von erhöhtem Interesse scheint, so muss dieser Umstand bereits während der Entwicklungszeit – also während der Modellierung – berücksichtigt werden. So muss es dem Modellierer beispielsweise möglich sein, Partitionen, die getrennt von einander evaluiert werden sollen, festzulegen.

- **Bewertung der Datenqualität**
  Daten, die von Sensoren oder anderen Quellen zur Verfügung gestellt werden, variieren zur Laufzeit sowohl in ihrer Qualität, als auch in ihrer Reichhaltigkeit an Informationen [SLP04].

  Die Bewertung der Qualität der Daten, die dem System zugrunde gelegt werden, findet sich auch in Anforderung A08 wieder. Sie fordert, dass jede Kontextinformation mit einer gewissen Unsicherheit belastet sein kann. Diese Unsicherheit quantifizieren zu können, ist für die korrekte Ausführung eines kontextsensitiven Systems von erhöhter Bedeutung, da daraus abgeleitet werden kann, mit welcher Wahrscheinlichkeit die aktuelle Situation des Nutzers korrekt bewertet werden kann.

- **Unvollständigkeit der Daten**
  Es muss für kontextsensitive Systeme davon ausgegangen werden, dass nicht alle Daten, von denen zur Entwicklungszeit angenommen wurde, sie wären zur Laufzeit verfügbar, auch tatsächlich verfügbar sind [SLP04].

  Diese Anforderung wurde ebenfalls in der durchgeführten Anforderungsanalyse ermittelt (vgl. Anforderung A11). Durch diese Anforderung wird verlangt, dass das durch das Modell beschriebene kontextsensitive System auch dann in Teilen funktioniert, wenn nicht alle modellierten Daten zur Laufzeit vorliegen (zum Beispiel weil bestimmte Werte aufgrund von Sensorausfällen nicht zur Verfügung stehen).

- **Einheitliches Datenverständnis**
  Strang et al. halten es außerdem für erforderlich, dass es Techniken zur modellbasierten Entwicklung kontextsensitiver Systeme erlauben, ein gemeinsames Verständnis für bestimmte Fakten aufzubauen. Alle Parteien eines kontextsensitiven Systems sollten sich über die Bedeutung ihrer geteilten Daten einig sein. [SLP04].

  Das Einfordern eines einheitlichen Datenverständnisses zeigt sich auch in Anforderung A12. Sie fordert, dass das Modell der Kontextinformationen als Austauschformat für verschiedene Prozesse, Disziplinen und Systeme dienen können muss. Um diese Anforderung zu erfüllen, muss das Modell die Möglichkeit bieten, ein für alle beteiligten Parteien einheitliches Datenverständnis bereit zu halten, weshalb diese Anforderung eng mit der von Strang et al. aufgestellten verbunden ist.

- **Integration in bestehende Systeme**
  Abschließend nennen Strang et al. das Kriterium der Integration in bestehende Systeme. Es beschreibt, wie leicht sich die Modellierung von Kontextinformationen mit bereits bestehenden Systemen verbinden lässt. [SLP04]

  Dass sich kontextsensitive Systeme leicht mit bereits bestehenden Bestandteilen des Systems *Fahrzeug* verbinden lassen sollen, wird auch durch Anforderung A01 gefordert. Auch wenn diese Anforderung insbesondere das System zur Laufzeit betrifft, so muss durch die Wahl der Modellierungstechnik sichergestellt werden, dass der Modellierer die Integration in bestehende Systeme zur Entwicklungszeit mit modellieren kann.

Auf Basis dieser Vergleichskriterien, untersuchten Strang et al. anschließend die folgenden Techniken zur Modellierung von Kontextinformationen:

- Schlüssel-Wert-Paare
- Profilbasierte-Modelle
- Grafische Modelle
- Objektorientierte Modelle
- Logikbasierte-Modelle
- Ontologiebasierte Modelle

Die Ergebnisse dieses Vergleichs sind in Tabelle 4.1 dargestellt. Während Schlüssel-Wert-Paare, Logikbasierte-Modelle, Profilbasierte-Modelle und Grafische Modelle bei diesem Vergleich schlecht bis mittelmäßig abschneiden, werden Objektorientierte Modelle und Ontologiebasierte Modelle durchweg positiv mit leichten Schwächen bei der Bewertung der Datenqualität, den Umgang mit unvollständigen Daten und die Integration in bestehende Systeme bewertet.

Zum von Strang et al. durchgeführten Vergleich zur Modellierung von Kontextinformationen lässt sich feststellen, dass sowohl die verglichenen Techniken zur Modellierung von Kontextinformationen, als auch die Anforderungen bzw. Vergleichskriterien eine große Überschneidung mit den im Rahmen dieser Arbeit untersuchten Techniken und Anforderungen aufweisen. Es wird daher davon ausgegangen, dass Ontologiebasierte Modelle und Objektorientierte Modelle auch für die Modellierung von Kontextinformationen in kontextsensitiven FIS geeignet sind.

4.1.2 Vergleich von Bettini et al.

Ein weiterer Vergleich von Techniken zur Modellierung von Kontextinformationen wurde 2010 von Bettini et al. vorgestellt [BBH+10]. Dabei wurden teilweise identische oder ähnliche Vergleichskriterien verwendet (*Verteilter Aufbau* sowie *Unvollständigkeit von Daten*). Ergänzend wurden bei diesem Vergleich auch die folgenden Kriterien berücksichtigt:

	Schlüssel-Wert-Paare	Logikbasierte Modelle	Profilbasierte Modelle	Grafische Modelle	Objektorientierte Modelle	Ontologie-basierte Modelle
Verteilter Aufbau	–	++	+	– –	++	++
Partielle Auswertung	–	–	++	–	+	++
Bewertung der Datenqualität	– –	–	–	+	+	+
Unvollständigkeit der Daten	– –	–	–	–	+	+
Formalität	– –	++	+	+	+	++
Einsatz in bestehenden Systemen	+	– –	++	+	+	+

Tabelle 4.1: Bewertung verschiedener Modelle nach Strang et al. [SLP04]

- **Unterstützung heterogener Daten**
  Laut Bettini et al. müssen kontextsensitive Systeme mit Daten umgehen können, die in sich ihrer Charakteristik mitunter stark unterscheiden [BBH+10].

  Auch in der im Rahmen dieser Arbeit durchgeführten Anforderungsanalyse wurde diese Anforderung identifiziert, wenngleich daraus zwei Anforderungen entstanden sind: Anforderung A04 fordert, dass ein kontextsensitives System und seine Modellierung verschiedene Datentypen berücksichtigen können muss. Anforderung A05 ergänzt, dass das System auch mit variierenden Frequenzen der Aktualisierungen von Daten umgehen können muss.

- **Abbilden von Beziehungen zwischen Daten**
  Bettini et al. vergleichen Techniken zur Modellierung von Kontextinformationen auch anhand der Möglichkeiten, Abhängigkeiten zwischen den modellieten Daten im Modell abbilden zu können [BBH+10].

  Die bereits vorgestellte Anforderung A02 fordert ähnlich, dass die Modellierung von Kontextinformationen Methoden zur Verfügung stellen muss, die es erlauben, Abstraktionen und Aggregationen zu modellieren. In einem Fahrzeug ist so beispielsweise das Beschleunigungsverhalten des Fahrzeugs von der Anzahl der Insassen, oder allgemeiner gesprochen, vom Beladungszustand des Fahrzeugs, abhängig.

- **Zeitbezug für Daten**
  Für ein kontextsensitives System ist es mitunter nicht nur von Interesse, welchen Wert ein bestimmtes Datum aktuell hat, sondern auch welchen Wert es in der Vergangenheit hatte, oder sogar welchen Wert ein Datum womöglich in der Zukunft haben wird [BBH+10].

  Da diese Anforderung in der durchgeführten Anforderungsanalyse nicht identifiziert wurde, wird die Fähigkeit zum Zeitbezug für eine Auswahl einer Technik zur Modellierung von Kontextinformationen als untergeordnet wichtig eingestuft.

- **Möglichkeiten zur Schlussfolgerung**
  Eine grundlegende Aufgabe kontextsensitiver Systeme besteht darin, ein Urteil über die aktuelle Situation seines Nutzers zu erstellen [BBH+10].

  Techniken zur Modellierung und Ausführung kontextsensitiver Systeme müssen daher die Möglichkeit bieten, aus bestehenden, modellierten Fakten, neue Fakten ableiten und schlussfolgern zu können. Da die Fähigkeit zur Schlussfolgern eine Möglichkeit zur Realisierung von Abstraktionen und Aggregationen ist, stellt diese Anforderung eine Spezialisierung der Anforderung A02 dar. Techniken, die die Möglichkeit zum Schlussfolgern bieten, bieten daher automatisch auch die Möglichkeit, Kontextinformationen zu abstrahieren und aggregieren, weshalb beide Anforderungen synonym betrachtet werden können.

- **Benutzbarkeit**
  Modelle kontextsensitiver Systeme sollen von möglichst vielen Menschen erstellt werden können. Der dem Modell zugrundeliegende Formalismus muss daher leicht verständlich sein, so dass Modellierer kontextsensitiver Systeme realweltliche Fakten schnell in diesen Formalismus überführen können [BBH+10].

  Anforderung A12 beschreibt, dass das Modell der Kontextinformation als Austausch-Medium benutzt werden können soll. Anforderung A19 fordert ergänzend, dass die Modellierung von Kontextinformationen auch von Personen benutzt werden können soll, die keine Software-Entwickler oder Programmierer sind. Ein hoher Grad an Benutzbarkeit wurde daher auch in der im Rahmen dieser Arbeit durchgeführten Anforderungsanalyse identifiziert.

- **Effiziente Bereitstellung von Daten**
  Aufgrund der Tatsache, dass kontextsensitive Systeme mit einer Vielzahl an Daten umgehen müssen, fordern Bettini et al. auch, dass Informationen und Daten in kontextsensitiven Systemen schnell und effizient abgerufen werden können [BBH+10].

  Die Forderung nach einer möglichst schnellen und effizienten Auswertung und Bereitstellung der Daten wurde in der durchgeführten Anforderungsanalyse nicht identifiziert. Da die vorliegende Arbeit sich auf FIS beschränkt und damit Fahrdynamisch-relevante Funktionen ausschließt, spielt die Effizienz nur eine untergeordnete Rolle. Nichtsdestotrotz ist es unabdingbar, die Geschwindigkeit der Auswertung zur Laufzeit zu bewerten.

Auf Basis dieser Vergleichskriterien führen Bettini et al. ebenfalls eine Bewertung durch und kommen dabei zu den in Tabelle 4.2 dargestellten Ergebnissen. Im Ver-

	Schlüssel-Wert-Paare	Profilbasierte Modelle	Grafische Modelle	Ontologie-basierte Modelle	Hybride Modelle
Verteilter Aufbau	–	–	○	–	○
Unvollständigkeit von Daten	–	–	○	–	+
Unterstützung heterogener Daten	–	–	+	+	+
Abbilden von Beziehungen zwischen Daten	–	–	○	+	+
Zeitbezug für Daten	–	–	+	–	+
Möglichkeiten zur Schlussfolgerung	–	–	○	+	+
Benutzbarkeit	○	○	+	○	○
Effiziente Bereitstellung von Daten	○	○	○	–	+

Tabelle 4.2: Bewertung verschiedener Modelle nach Bettini et al. [BBH+10]

gleich zu den Ergebnissen von Strang et al. fällt auf, dass Ontologiebasierte Modelle nicht in jeder Kategorie positiv bewertet werden. So werden ihre Fähigkeiten mit Bezug auf die Kriterien *Verteilter Aufbau*, *Unvollständigkeit von Daten* und *Effiziente Bereitstellung von Daten* als negativ beurteilt. Ontologiebasierte Modelle weisen laut Bettini et al. außerdem lediglich eine mittelmäßige Benutzbarkeit auf. Zusammenfassend kommen Bettini et al. nach ihrem Vergleich zu dem Schluss, dass keines der bis dahin existierenden Techniken zur Modellierung kontextsensitiver Systeme vollends überzeugen kann. Sie schlagen daher vor, sogenannten *Hybride Modelle* zu verwenden, welche bestehende Techniken miteinander kombinieren, um Vorteile zu maximieren und Nachteile einzelner Techniken zu minimieren oder gar zu eliminieren.

4.1.3 Vergleich von Bergmann

In einem subsumierenden Vergleich ergänzt Bergmann weitere Kriterien für einen Vergleich verschiedener Techniken zur Modellierung kontextsensitiver Systeme [Ber14].

- **Unterstützung einer Regel-basierten Auswertung**
  Bergmann fordert, dass Modellierer durch die verwendete Modellierungstechnik die Möglichkeit geboten bekommen müssen, die Auswertung des Modells ihren gewohnten Denkweisen entsprechend kontrollieren zu können. Bergmann kommt dabei zu dem Schluss, dass eben diese gewohnte Denkweise Regel-basiert abläuft. Nach Bergmann ist es daher wichtig, dass Techniken zur Modellierung

kontextsensitiver Systeme die Möglichkeit bieten, die Auswertung auf Basis von Regeln durchführen zu könnenn [Ber14].

Die Anforderung Fakten und Wissen durch Regeln im Modell der Kontextinformationen zu hinterlegen, wurde auch in der im Rahmen dieser Arbeit durchgeführten Anforderungsanalyse ermittelt (vgl. Anforderung A03). Auch bei dieser Analyse gaben die befragten an, dass Regeln benötigt werden, weil das Wissen der Modellierer häufig in Form von Regeln formuliert wird.

- **Unterstützung von Unsicherheiten**
  Erstmalig wird von Bergmann festgestellt, dass Kontextinformationen mit einer individuellen Unsicherheit belegt sein können. Daher wird die Fähigkeit, diese Unsicherheiten modellieren zu können, als Vergleichskriterium für den durchgeführten Vergleich aufgenommen [Ber14].

  Anforderung A08 forderte, dass das Modell der Kontextinformationen Möglichkeiten bieten muss, für jede Quelle von Kontextinformationen hinterlegen zu können, mit welcher Unsicherheit die jeweilige Information belegt ist. Diese Anforderung deckt sich daher mit der von Bergmann aufgestellten Anforderung zur Unterstützung von Unsicherheiten.

- **Standardisierung**
  Ob für die jeweiligen Techniken zur Modellierung kontextsensitiver Systeme standardisierte Methoden, Sprachen oder Programmierschnittstellen existieren, ist für Bergmann im Hinblick auf die industrielle Tauglichkeit ebenfalls von erhöhter Bedeutung [Ber14] und wird daher als Vergleichskriterium hinzugefügt.

  Zur Minimierung des Entwicklungs- und Wartungsaufwandes wurde die Anforderung nach einer standardisierten Möglichkeit zur Modellierung von Kontextinformationen auch in der durchgeführten Anforderungsanalyse ermittelt (vgl. Anforderung A20). Auch diese von Bergmann aufgestellte Anforderung deckt sich daher mit denen der vorliegenden Arbeit.

- **Tool-Unterstützung**
  Ebenfalls für die industrielle Tauglichkeit entscheidend ist die Quantität und Qualität verfügbarer Tools für die jeweiligen Techniken zur Modellierung kontextsensitiver Systeme. Bergmann definieren die Tool-Unterstützung daher als zusätzliches Kriterium für den Vergleich [Ber14].

  Anforderung A21 forderte ebenfalls ein Vorhandensein von Tools zur Erstellung des jeweiligen Modells, um die Modellierer bei der Modellierung bestmöglich unterstützen zu können und Fehler bei der Modellierung zu minimieren.

- **Stand der Technik**
  Mit diesem abschließenden Kriterium möchte Bergmann die Aktualität und den Grad der Akzeptanz der jeweiligen Technik zur Modellierung kontextsensitiver Systeme bewerten. Als Grundlage zur Bewertung diente dabei eine umfangreiche Literaturrecherche [Ber14].

  Anforderung A22 zeigt, dass die Aktualität der Technik zur Modellierung von Kontextinformationen auch bei der Erstellung kontextsensitiver FIS eine entscheidende Rolle spielt.

	Schlüssel-Wert-Paare	Logikbasierte Modelle	Profilbasierte Modelle	Grafische Modelle	Objektorientierte Modelle	Ontologie-basierte Modelle	Hybride Modelle
Unterstützung heterogener Daten	–	–	○	+	+	+	+
Unterstützung einer Regel-basierten Auswertung	–	+	–	+	–	+	+
Abbilden hierarchischer Strukturen zwischen Daten	–	–	○	○	+	+	+
Unterstützung von Unsicherheiten & Unvollständigkeiten der Daten	–	–	–	○	–	–	+
Standardisierung	–	–	+	○	○	+	○
Tool-Unterstützung	–	–	–	–	–	+	○
Effizienz	+	○	○	○	○	–	○
Stand der Technik	–	–	–	–	+	+	+

Tabelle 4.3: Bewertung verschiedener Modelle nach Bergmann [Ber14]

Die Ergebnisse des von Bergmann durchgeführten Vergleichs sind in Tabelle 4.3 aufgetragen. Auch dieser Vergleich zeigt, das Ontologiebasierte Modelle Schwächen in den Bereichen der Effizienz und dem Umgang mit unsicheren Daten aufweisen. Außerdem, werden, wie bereits von Bettini et al. (vgl. Abschnitt 4.1.2), Hybride Modelle von Bergmann in allen Bereichen positiv bewertet.

4.1.4 Fazit

Basierend auf den vorgestellten Vergleichen von Strang et al. [SLP04], Bettini et al. [BBH+10] und Bergmann [Ber14] zu verschiedenen Techniken zur Modellierung kontextsensitiver Systeme, wird im Rahmen dieser Arbeit davon ausgegangen, dass Ontologiebasierte Modelle die derzeit beste Basis zur modellbasierten Entwicklung kontextsensitiver Systeme darstellen. Da Objektorientierte Modelle ebenfalls überwiegend positiv bewertet wurden und Bettini et al. Hybride Modelle für vielversprechend halten [BBH+10], wird im weiteren Verlauf gezeigt, wie Ontologiebasierte Modelle und Objektorientierte Modelle kombiniert werden können, um kontextsensitive FIS modellbasiert zu entwickeln. Dabei wird insbesondere darauf eingegangen, wie Unsi-

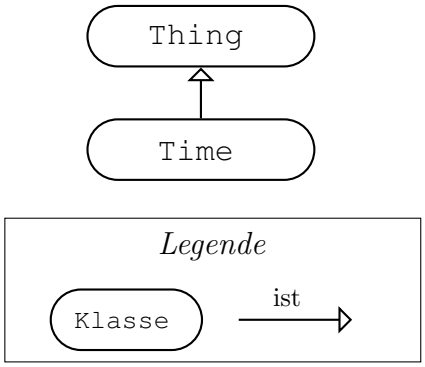

Abbildung 4.1: Exemplarische Ontologie mit einer Klasse `Time`

cherheiten, Unvollständigkeiten und der Umgang mit redundanten Datenquellen von Kontextinformationen modelliert werden können.

4.2 Modellierung durch Ontologien

In diesem Abschnitt wird gezeigt, wie Ontologiebasierte Modelle um Objektorientierte Ansätze erweitert werden können, um kontextsensitive FIS Model-basiert zu entwickeln. Dabei werden Ontologien mit den aus der objektorientierten Programmierung bekannten Konzepten der *Abstraktion*, der *Vererbung* und der *Annotationen* erweitert. Zunächst wird dazu in Abschnitt 4.2.1 gezeigt, wie die in einem Fahrzeug verfügbaren Kontextinformationen in der Ontologie abgebildet werden, damit sie anschließend, wie in Abschnitt 4.2.2 beschrieben, zu höherwertigen Kontextinformationen abstrahiert und aggregiert werden können. Da Ontologien bisher lediglich unzureichende Möglichkeiten bieten, Unsicherheiten und Daten aus multiplen Quellen zu behandeln, wird in Abschnitt 4.2.3 und Abschnitt 4.2.4 beschrieben, welche Erweiterungen im Rahmen dieser Arbeit vorgeschlagen werden, um diese Unzulänglichkeiten zu eliminieren. In Abschnitt 4.2.5 wird abschließend erläutert, wie der Modellierer zur Entwicklungszeit durch den Einsatz von Annotationen beeinflussen kann, wie sich das dem Modell entsprechende System zur Laufzeit im Detail verhält.

4.2.1 Abbilden verfügbarer Kontextinformationen

Alle relevanten Kontextinformation, die von dem kontextsensitiven FIS verarbeitet werden, werden in einem ersten Schritt als Klassen innerhalb der Ontologie abgebildet. Dabei ist es zunächst nicht von Bedeutung, welche Komponente diese Information später tatsächlich zur Verfügung stellt, oder ob diese Information unter Umständen sogar von verschiedenen Komponenten zur Verfügung gestellt wird. So kann beispielsweise wie in Abbildung 4.1 gezeigt, die Kontextinformation einer Uhrzeit, als Klasse `Time` in der Ontologie angelegt werden. Wie jede Klasse einer Ontologie, ist `Time` dabei als Subklasse der Klasse `Thing` dargestellt.

Data Properties

Jeder der so angelegten Klassen werden anschließend sogenannte *Data Properties* zugewiesen, welche die eigentlichen Daten der Kontextinformation enthalten. Die Kontextinformation `Time` könnte so beispielsweise die Data Properties `hour`, `minute`

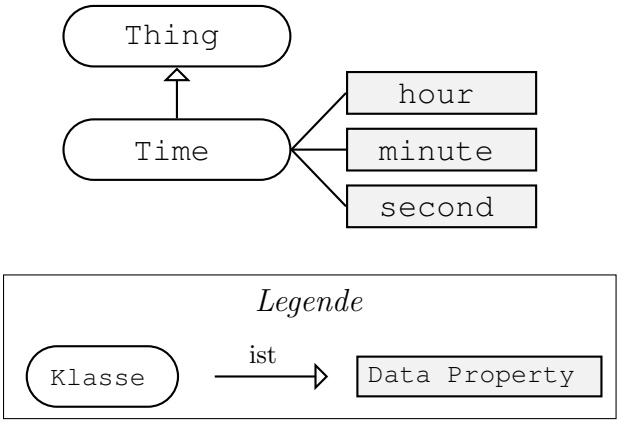

Abbildung 4.2: Exemplarische Ontologie mit einer Klasse `Time` und drei dazugehörigen Data Properties `hour`, `minute` und `second`

und `second` zugewiesen bekommen, um die Uhrzeit sekundengenau abbilden zu können. Eine um diese Data Properties erweiterte Ontologie ist in Abbildung 4.2 dargestellt. Bei der Definition der Data Properties kann sowohl auf alle einfachen OWL-Datentypen[1], als auch auf eigene, zuvor definierte, komplexe Datentypen zurückgegriffen werden. Außerdem können bei der Zuweisung von Data Properties zu Klassen Kardinalitäten verwendet werden, um Aussagen über die Quantität dieser Beziehung zu ermöglichen. Dadurch wird es prinzipiell ermöglicht, dass einer Kontextinformation (Klasse einer Ontologie) mehrere Daten (Data Properties einer Ontologie) des gleichen Typs zugrunde liegen können.

Auch wenn Data Properties bereits fester Bestandteil von Ontologien sind, so ist ihre Verwendung im Rahmen dieser Arbeit deutlich Objektorientierter als ursprünglich vorgesehen. Während Data Properties in Ontologien bisher eigenständig angelegt und über Klassen-Beziehungen einer oder mehreren Klassen zugewiesen wurden, so werden sie in dem hier vorgestellten Ansatz wie Variablen eines Objekts verwendet und pro definierter Klasse festgelegt.

Subklassen

Der zweite, im Rahmen dieser Arbeit vorgeschlagene, Schritt zur Modellierung kontextsensitiver FIS durch Ontologien, stellt die Verwendung von Subklassen dar. Diese können, wie aus der Objektorientierten Programmierung bekannt, eingesetzt werden, um Gemeinsamkeiten bestimmter Klassen an nur einer Stelle modellieren zu müssen. Die Kontextinformationen `CurrentTime` (zur Repräsentation der aktuellen Uhrzeit) und `JourneyTime` (zur Repräsentation der gefahrenen Zeit) könnten beispielsweise, wie in Abbildung 4.3 dargestellt, als Subklassen der Klasse `Time` modelliert werden und würden so die für `Time` angelegten Data Properties erben.

Darüber hinaus können Subklassen auch dazu verwendet werden, einzelne Kontextinformationen zu klassifizieren, um die Weiterverarbeitung zu vereinfachen. Die Kontextinformation `JourneyTime` könnte so beispielsweise durch Anlegen der Subklassen `Short`, `Medium` und `Long`, wie in Abbildung 4.4 gezeigt, in drei weitere

[1] Online abrufbar unter: http://www.w3.org/TR/owl2-syntax/#Datatype_Maps. Zuletzt überprüft am 19.10.2016.

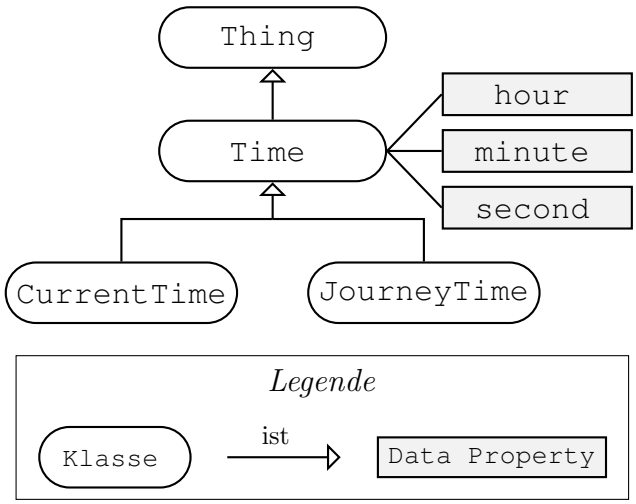

Abbildung 4.3: Exemplarische Ontologie mit den zwei Subklassen `CurrentTime` und `JourneyTime`

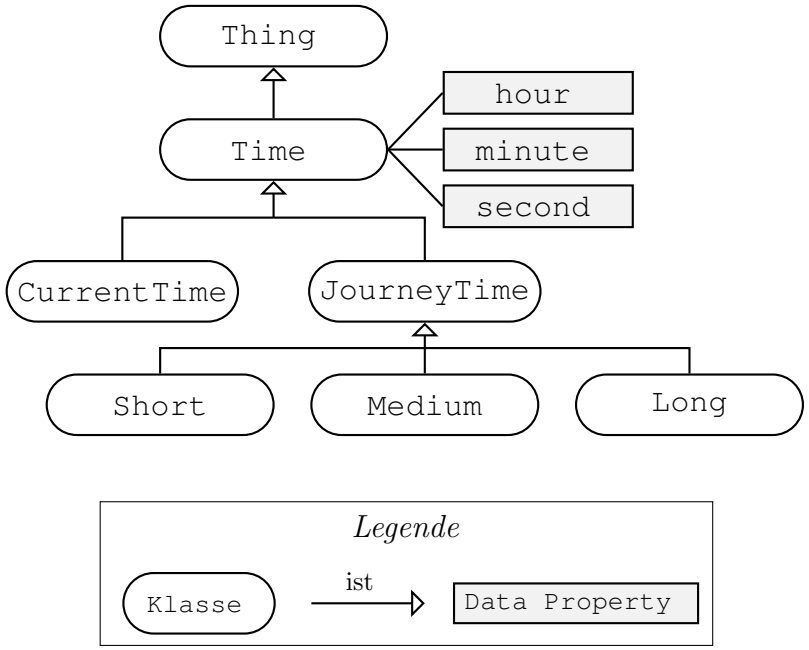

Abbildung 4.4: Exemplarische Ontologie mit den Subklassen `Short`, `Medium` und `Long` zur Klassifizierung der aktuellen Fahrtdauer

	Datenbereich für `Short`	Datenbereich für `Medium`	Datenbereich für `Long`
Scharfe Trennung der Subklassen	$0 \leq \texttt{hour} \leq 1$	$2 \leq \texttt{hour} \leq 3$	$4 \leq \texttt{hour}$
Unscharfe Trennung der Subklassen	$0 \leq \texttt{hour} \leq 2$	$2 \leq \texttt{hour} \leq 4$	$4 \leq \texttt{hour}$

Tabelle 4.4: Exemplarische scharfe und unscharfe Trennung von Subklassen am Beispiel der Zerlegung einer Fahrtzeit in kurz, mittel und lang

Subklassen aufgeteilt werden. Individuen können zur Laufzeit dann nicht nur der Klasse `JourneyTime` angehören, sondern zum Beispiel auch der Klasse `Long`, und so repräsentieren, dass die aktuelle Fahrtzeit als *lang* einzustufen ist. Dabei sei erneut auf die Besonderheit der *Open World Assumption* hingewiesen (vgl. Abschnitt 2.2.6), die es prinzipiell erlaubt, dass ein Individuum in mehr als einer Klasse gleichzeitig sein kann. Ein Individuum kann so beispielsweise gleichzeitig in den Klassen `Medium` und `Long` sein. Während dieser Umstand für einige Anwendungsfälle hilfreich sein kann (etwa bei der Modellierung unscharfer Übergänge), so ist bei anderen Szenarien eine scharfe Trennung erwünscht. Soll beispielsweise die Sitzplatzbelegung eines Fahrzeug ausgewertet werden und diese dafür in die beiden Subklassen `AloneInCar` (zur Repräsentation, dass der Fahrer allein im Fahrzeug sitzt) und `NotAloneInCar` (zur Repräsentation, dass der Fahrer *nicht* allein im Fahrzeug sitzt) klassifiziert werden, scheint eine Zugehörigkeit zu beiden Klassen gleichzeitig wenig hilfreich. Der Modellierer muss daher auf die dem Anwendungsfall entsprechende Zuweisung von Subklassen achten. Welche Sprachmittel dafür zur Verfügung stehen, zeigen die nächsten beiden Abschnitte.

Datenbereiche

Eine Möglichkeit, die Zugehörigkeit von Individuen zu Klassen zu kontrollieren, stellen Datenbereiche dar. Ihre Verwendung erlaubt es, Klassen nur dann einer bestimmten (Sub-)Klasse zuzuweisen, wenn die Werte seiner Data Properties in einem bestimmten Bereich liegen. Ein Individuum der Klasse `JourneyTime` mit den Data Properties `hour`, `minute` und `second` (geerbt von `Time`), könnte so beispielsweise nur dann der Subklassen `Long` zugewiesen werden, wenn die Data Property `hour` einen Wert größer 4 ($\texttt{hour} > 4$) angenommen hat. Abbildung 4.5 zeigt die um Datenbereiche ergänzte Ontologie. Je nachdem, wie der Modellierer die Datenbereiche der anderen Subklassen `Short` und `Medium` auslegt, kann so eine scharfe oder unscharfe Trennung der Subklassen erreicht werden. Tabelle 4.4 zeigt exemplarisch, wie die Datenbereiche dafür entsprechend definiert werden können.

SWRL-Regeln

Eine weitere Möglichkeit, Individuen zu klassifizieren, bieten die sogenannten *SWRL-Regeln* (vgl. Abschnitt 2.2.6). Die Verwendung von SWRL-Regeln erlaubt es dem Modellierer, realweltliche Fakten regelbasiert abzubilden. Sie eignen sich auch für die einfache Klassifikation von Subklassen. So können, alternativ zur Verwendung von

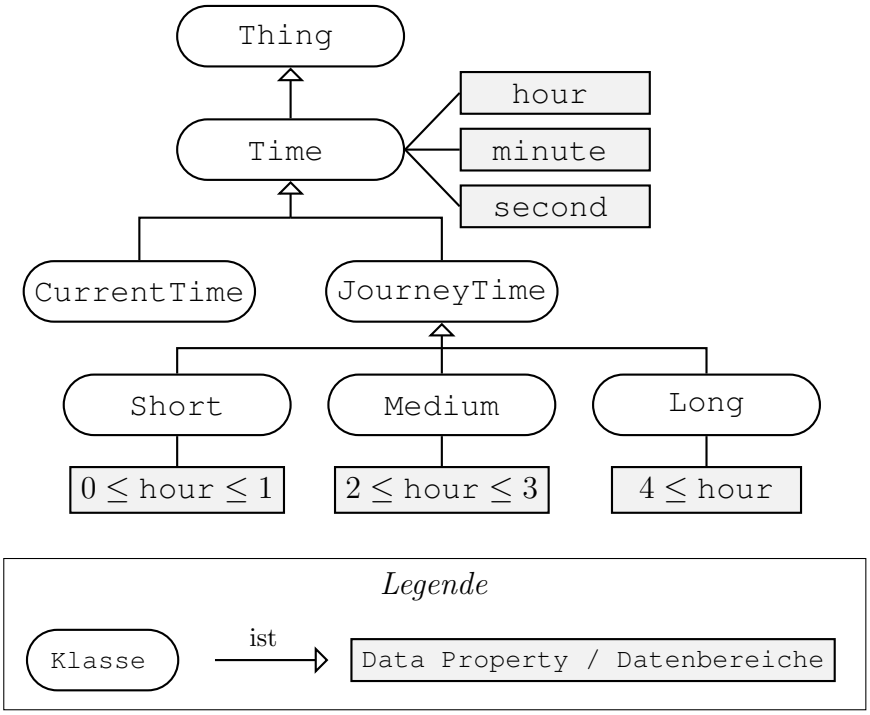

Abbildung 4.5: Exemplarische Ontologie mit Datenbereichen an den Klassen `Short`, `Medium` und `Long`

Data Properties, Individuen der Klasse `JourneyTime` durch die Verwendung der SWRL-Regel 4.1 als `Medium` klassifiziert werden.

SWRL-Regel 4.1: *Exemplarische Klassifikation eines Individuums der Klasse* `JourneyTime` *in die Klasse* `Medium` *in Abhängigkeit der Data Property* `hour`

$$\texttt{j} \in \texttt{JourneyTime} \land \texttt{j.hour} \geq 2 \land \texttt{j.hour} \leq 3 \Rightarrow \texttt{j} \in \texttt{Medium}$$

Durch die Verwendung von Klassen, Subklassen, Datenbereichen und SWRL-Regeln kann die reale Welt Stück für Stück vom Modellierer nachgebildet werden. Dabei ist es jedoch nicht immer erforderlich, alle Fakten zu modellieren, sondern lediglich diejenigen, die für den konkreten Anwendungsfall von entscheidender Relevanz sind.

SWRL-Regeln dienen jedoch nicht ausschließlich der Klassifikation von Subklassen, sondern sind auch ein mächtiges Werkzeug zur Modellierung *höherwertiger Kontextinformationen*, wie der nächste Abschnitt zeigen wird.

4.2.2 Modellieren höherwertiger Kontextinformationen

In einem zweiten Schritt kann dann aus den zuvor modellierten Kontextinformationen die Zusammensetzung *höherwertiger Kontextinformationen* modelliert werden. So kann beispielsweise aus den Kontextinformationen `JourneyTime` und `CurrentTime` die höherwertige Kontextinformation `Drowsiness` abgeleitet werden, die beschreibt, ob sich beim Nutzer eine Tendenz zur Müdigkeit erkennen lässt. Auch diese höherwertigen Kontextinformationen werden in der Ontologie

als Klassen abgebildet. Gleichzeitig wird dem Modellierer die Möglichkeit geboten, höherwertige Kontextinformationen je nach Ausprägung der zugrundeliegenden Kontextinformationen in verschiedene Subklassen zu unterteilen. Die Kontextinformation `Drowsiness` könnte so beispielsweise je nach Ausprägung der zugrundeliegenden Kontextinformationen als eine der Subklassen `No`, `Medium` oder `High` klassifiziert werden (siehe Abbildung 4.6). Diese Klassifikationen können vom Modellierer ebenfalls durch Subklassen oder SWRL-Regeln modelliert werden. Bei der Klassifikation durch SWRL-Regeln hat der Modellierer dabei die Möglichkeit zu überprüfen, ob Individuen der zugrundeliegenden Kontextinformationen zur Laufzeit bereits in eine ihrer Subklassen klassifiziert wurden. So kann er durch die Verwendung der SWRL-Regel 4.2 beispielsweise definieren, dass `Drowsiness` immer als `High` klassifiziert wird, wenn die `JourneyTime` als `Long` klassifiziert wurde.

SWRL-Regel 4.2: *Exemplarische Klassifikation eines Individuums der Klasse* `Drowsiness` *in die Klasse* `High` *in Abhängigkeit des Vorhandenseins eines Individuums der Klasse* `Long`

$$\texttt{j} \in \texttt{JourneyTime} \land \texttt{j} \in \texttt{Long} \land \texttt{d} \in \texttt{Drowsiness} \Rightarrow \texttt{d} \in \texttt{High}$$

Solche Beziehungen zwischen Klassen erlauben es dem Modellier, auf Basis von Kontextinformationen ein *Situationsverständnis* zu modellieren. So können abstrakte Situation wie *Gestresster Fahrer*, *Gefährliche Fahrsituation* oder *Pendelfahrt zur Arbeit* als Situationen in der Ontologie abgebildet werden. Individuen können basierend auf einfachen Kontextinformationen als solche Situationen klassifiziert werden und damit zur Laufzeit signalisieren, dass die jeweils aktuellen Kontextinformationen darauf schließen lassen, dass sich der Nutzer aktuell in einer dieser Situationen befindet.

4.2.3 Behandlung von Unsicherheiten

Eine im Rahmen dieser Arbeit vorgenommene Erweiterung am Ontologie-basierten Ansatz zur Modellierung kontextsensitiver FIS beschreibt die Behandlung von *unsicheren Kontextinformationen*. Die grundlegende Annahme für die Notwendig einer Behandlung von Unsicherheiten ist, dass jede im System verfügbaren Kontextinformation eine gewisse Sicherheit bzw. Unsicherheit (engl. *Confidence*) zugeschrieben werden kann. Diese gibt jeweils an, mit welcher Sicherheit der aktuelle Wert der Kontextinformation als *wahr* angenommen werden kann (vgl. Anforderung A08 und Anforderung A09 aus Kapitel 3). Die Problemstellung, mit der sich die Behandlung von Unsicherheiten auseinandersetzt lautet: Welche Sicherheit hat eine höherwertige Kontextinformation, der mehrere, unsichere Kontextinformationen zugrunde liegen?

Zur Beantwortung dieser Problemstellung wurden Ontologien im Rahmen dieser Arbeit um das Konzept des *Wahrheitswertes* ($\text{conf} \in [0, 1]$) ergänzt. Der Wahrheitswert gibt an, mit welcher Wahrscheinlichkeit der Wert einer bestimmten Kontextinformation als wahr angenommen werden kann. So gibt $conf(Temperature) = 0.8$ beispielsweise an, dass der Wert der Kontextinformation *Temperatur* mit einer Wahrscheinlichkeit von 80% als wahr angenommen werden kann und somit eine

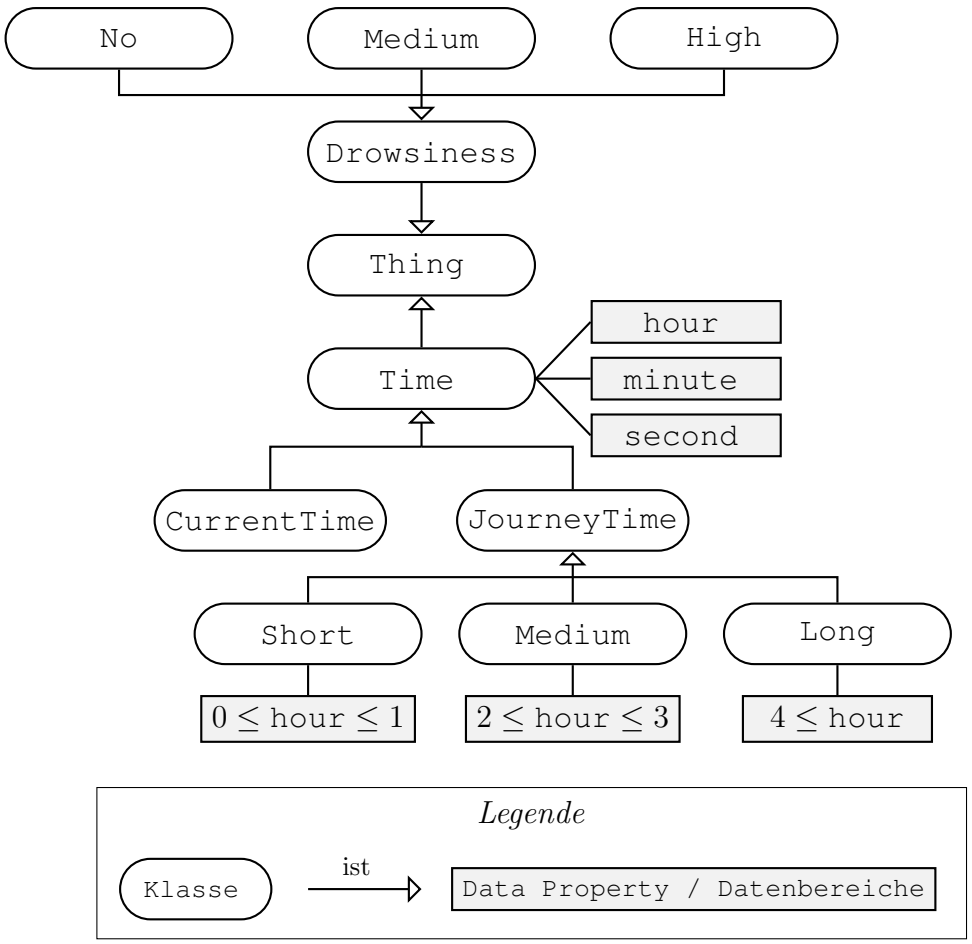

Abbildung 4.6: Exemplarische Ontologie mit der Klasse `Drowsiness` und ihren Subklassen `No`, `Medium` und `High` zur Klassifizierung der Müdigkeit des Fahrers

Norm zur Berechnung	t-Norm $a \wedge b$	t-Conorm $a \vee b$
Produkt	$a \cdot b$	$a + b - a \cdot b$
Łukasiewicz	$max\{a, b\}$	$min\{a, b\}$
Zadeh	$min\{a, b\}$	$max\{a, b\}$

Tabelle 4.5: Erklärung der verschiedenen Berechnungsnormen zur Kombination von Zahlenwerten

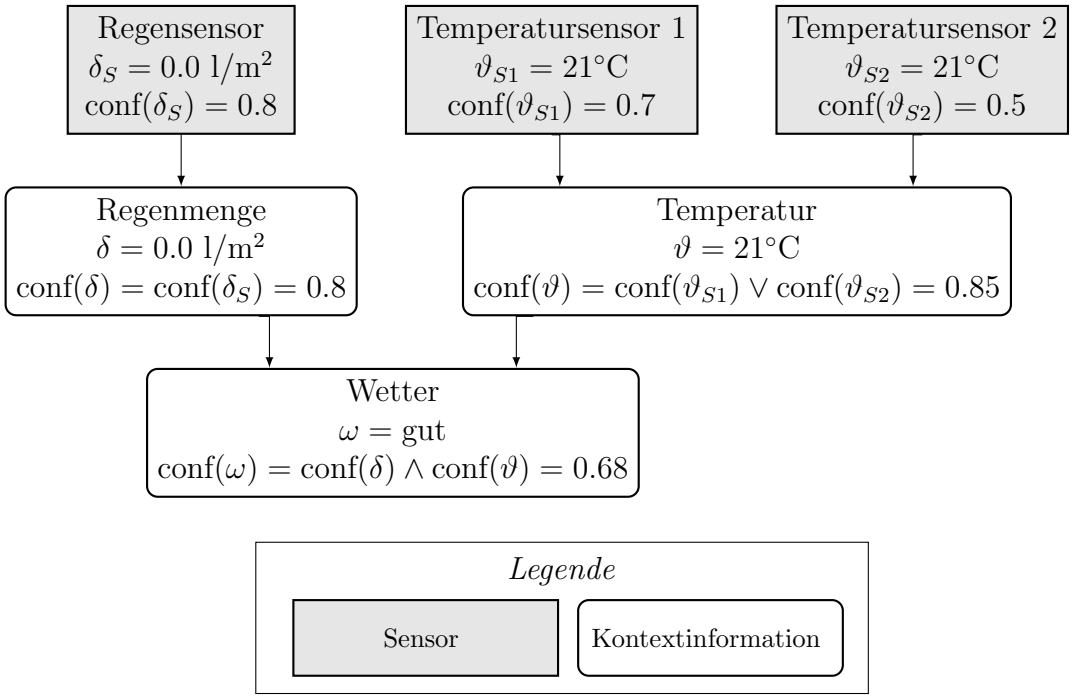

Abbildung 4.7: Exemplarische Berechnung der Wahrheitswerte anhand der Produktnorm

Unsicherheit von 20% verbleibt, dass der von der Quelle der Kontextinformation zur Verfügung gestellte Wert der Temperatur nicht korrekt ist.

Werden mehrere Kontextinformationen miteinander kombiniert, um eine höherwertige Kontextinformation daraus abzuleiten, wird die Berechnung des Wahrheitswert dieser höherwertigen Kontextinformation nach den aus der Fuzzy-Logik [Kru11] bekannten Berechnungsnormen durchgeführt. Werden semantisch verschiedenen Kontextinformationen miteinander kombiniert, werden t-Normen angewandt, während bei einer Kombination gleichbedeutender Kontextinformationen t-Conormen zum Einsatz kommen. Tabelle 4.5 zeigt eine Auswahl der aus der Fuzzy-Logik bekannten Berechnungsnormen sowie die dazugehörigen t-Normen und t-Conormen.

Während der Modellierung stehen dem Modellierer diese Normen zur Verfügung, um zu bestimmen, wie mit Unsicherheiten von Kontextinformationen zur Laufzeit umgegangen werden soll.

Rechenbeispiel

Das in Abbildung 4.7 dargestellte Beispiel zeigt, in welchen Fällen die t-Norm und in welchen Fällen die t-Conorm verwendet wird. Für diese exemplarische Berechnung der Wahrheitswerte sei angenommen, dass dem kontextsensitiven System ein Regensensor und zwei Temperatursensoren zur Verfügung stehen. Weiterhin sei angenommen, dass anhand der Regenmenge und der Temperatur die höherwertige Kontextinformation *Wetter* abgeleitet werden kann. Die drei Sensoren liefern Werte über die Regenmenge bzw. die aktuelle Temperatur mit einem bestimmten Wahrheitswert. Der vom Regensensor ermittelte Wert der aktuellen Regenmenge verfügt über einen Wahrheitswert von conf$(\delta_S) = 0.8$, der Temperatursensor 1 misst mit einem Wahrheitswert von

$\text{conf}(\vartheta_{S1}) = 0.7$ und die vom Temperatursensor 2 gemessen Temperatur entspricht mit einer Wahrscheinlichkeit von $\text{conf}(\vartheta_{S2}) = 0.5$ der tatsächlichen Temperatur. Aus diesen drei Sensoren werden zunächst zwei Kontextinformationen generiert: die Regenmenge und die Temperatur.

Während die Berechnung des Wahrheitswertes für die Regenmenge δ trivial ausfällt, weil der Wahrheitswert der Datenquelle übernommen werden kann, muss der Wahrheitswert für die Temperatur errechnet werden. Die intuitive Annahme für die Berechnung eines Wahrheitswertes wäre, dass sich der Wahrheitswert einer Kontextinformation erhöht, wenn mehrere Datenquellen den gleichen Wert melden. Da beide Temperatursensoren den gleichen Wert von 21°C melden, wird zur Berechnung des Wahrheitswertes der Kontextinformation Temperatur die t-Conorm verwendet: $\text{conf}(\vartheta) = \text{conf}(\vartheta_{S1}) \vee \text{conf}(\vartheta_{S2})$. Für dieses Beispiel sei angenommen, dass der Modellierer modelliert hat, dass zur Berechnung des Wahrheitswertes der Kontextinformation Temperatur die Produktnorm verwendet wird, weshalb sich die Berechnung des Wahrheitswertes gemäß Tabelle 4.5 zu $\text{conf}(\vartheta) = \text{conf}(\vartheta_{S1}) + \text{conf}(\vartheta_{S2}) - \text{conf}(\vartheta_{S1}) \cdot \text{conf}(\vartheta_{S2}) = 0.7 + 0.5 - 0.7 \cdot 0.5 = 0.85$ ergibt. Der Wahrheitswert der Kontextinformation liegt der intuitiven Annahme entsprechend also höher als der, der zugrundeliegenden Datenquellen, da die beiden Datenquellen die gleiche Information zur Verfügung gestellt haben und diese sich daher gegenseitig verstärken.

Bei der höherwertigen Kontextinformation *Wetter* werden die Kontextinformationen *Regenmenge* und *Temperatur* herangezogen. Da es sich hierbei um zwei verschiedene Kontextinformationen handelt, wird bei der Berechnung des Wahrheitswertes die t-Norm herangezogen: $\text{conf}(\omega) = \text{conf}(\delta) \wedge \text{conf}(\vartheta)$. Auch hier sei wieder angenommen, dass der Modellierer die Verwendung der Produktnorm modelliert hat, weshalb sich der Wahrheitswert gemäß Tabelle 4.5 zu $\text{conf}(\omega) = \text{conf}(\delta) \cdot \text{conf}(\vartheta) = 0.8 \cdot 0.85 = 0.68$ errechnet. Der Wahrheitswert der Kontextinformation fällt damit geringer aus, als der, der zugrundeliegenden Kontextinformationen. Auch das entspricht erneut der intuitiven Annahme, da die Verwendung zweier unsicherer Informationen zur Generierung einer neue Information einen negativen Einfluss auf die Sicherheit der generierten Informationen haben muss.

4.2.4 Kontextinformationen aus multiplen Quellen

Eine weitere in Kapitel 3 ermittelte Anforderung machte deutlich, dass Kontextinformationen mitunter mehrere, redundante Datenquellen aufweisen können (vgl. Anforderung A07). Im Rahmen dieser Arbeit wurde die Ontologie-basierte Modellierung kontextsensitiver Systeme daher um die Möglichkeit erweitert, dass eine Kontextinformation mehrere Quellen haben kann. Auch wenn die so geschaffenen Erweiterung insbesondere Auswirkungen auf die Implementierung des Systems haben, so hat der Modellierer bereits während der Modellierung Möglichkeiten den Umgang mit multiplen Quellen zu modellieren. Die Problemstellung, welche durch diese Erweiterung beantwortet werden soll lautet: Welche Werte haben die Daten der Kontextinformation, die von mehreren Quellen bereitgestellt wird?

Zur Beantwortung dieser Fragestellung wurde im Rahmen dieser Arbeit die Möglichkeit geschaffen, eine bestimmte Datenquellen mit einem bestimmten *Gewicht*

zu kennzeichnen. Auch wenn das tatsächliche Gewicht einer Datenquelle für den Modellierer vollkommen transparent bleibt, da die Datenquelle als solche nicht Teil des Modells ist, so bekommt er beispielsweise die Möglichkeit, bei der Modellierung der Kontextinformationen festzulegen, dass immer die Datenquelle mit dem höchsten Gewicht verwendet wird. Auch der Wahrheitswert einer Datenquelle kann genutzt werden, um eine Datenquelle festzulegen, die als Quelle für eine Kontextinformation genutzt werden soll.

Eine zweite Möglichkeit wird dem Modellierer in Form einer *Wertekombination* geboten. So kann der Modellierer beispielsweise festlegen, dass, sofern mehrere Datenquellen für eine Kontextinformation existieren und es sich um einen numerischen Wert handelt, die Kontextinformation den Mittelwert aller Datenquellen als Wert erhält. Auch gewichtete Mittelwerte unter Zuhilfenahme des Gewichts oder des Wahrheitswertes einer Datenquelle stehen dem Modellierer dabei zu Verfügung.

Wie alle anderen Informationen auch, werden der Umgang mit multiplen Quellen durch Annoationen an die entsprechenden modelliert. Wie genau und welche Annotationen dazu zum Einsatz kommen, zeigt der nächste Abschnitt.

4.2.5 Annotieren von Klassen

Nachdem Kontextinformationen und Situationen innerhalb der Ontologie modelliert wurden, müssen abschließend noch einige Metainformationen hinzugefügt werden, damit das Modell genutzt werden kann, um zur Laufzeit eine Aussage über bestimmte Situationen auf Basis einzelner Kontextinformationen zu machen. Dazu zählen insbesondere Informationen über die Quelle der Daten und deren Verknüpfung mit den Klassen und Data Properties der Ontologie, sowie der Umgang mit Redundanten Datenquellen und deren Wahrscheinlichkeiten.

Da diese Informationen pro vorhandener Klasse in der Ontologie relevant sind, werden sie als *Annotationen* einer Klasse innerhalb der Ontologie abgelegt. Die Annotationen helfen zum einen, die Verknüpfung zwischen Modell und Fahrzeug zu definieren, und zum anderen, wichtige Metainformationen an die Kontextinformationen anzufügen, um sie zur Laufzeit den Anforderungen entsprechend verwerten zu können.

Richtung des Datenaustausches – die ExchangeType Annotation

Mithilfe dieser Annotation wird festgelegt, ob eine Klasse zur Laufzeit einfache Kontextinformationen repräsentiert (*Exchange_In*), ob sie höherwertige Kontextinformationen, die aufgrund modellierter Zusammenhänge zwischen einfachen Kontextinformationen erschlossen wurden, für nachgelagerte Systeme zur Verfügung stellt (*Exchange_Out*), ob sie beide Funktionen gleichzeitig übernimmt (*Exchange_Both*) oder ob die Klasse weder Eingabe noch Ausgabe ist (*Exchange_None*), da sie beispielsweise lediglich strukturierende Aufgaben übernimmt, um die Übersichtlichkeit des Modells zu gewährleisten.

Name der Datenquelle – die SourceName Annotation

Diese Annotation erlaubt es, die im Modell durch Klassen abgebildeten Kontextinformationen einer realen Datenquelle zuzuweisen. Die Zuweisung geschieht dabei

über eine eindeutige Zeichenfolge. So kann beispielsweise signalisiert werden, dass eine bestimmte Kontextinformation (zum Beispiel die Außentemperatur) auf einem bestimmten CAN-Bus zur Verfügung steht. Der Name der Quelle, und damit der Inhalt dieser Annotation, wäre in diesem Fall dann beispielsweise *CAN-1*.

Zuordnung von Data Properties – die SourcePropertyName Annotation

Damit die *Data Properties*, die den Klassen während der Modellierung zugeordnet wurden, zur Laufzeit mit realen Daten versorgt werden können, wird mit Hilfe dieser Annotation eine Abbildung zwischen den Namen der *Data Properties* und den Namen der Daten innerhalb der entsprechenden Datenquelle erstellt. Im Falle der Außentemperatur könnte es sich dabei beispielsweise um die konkrete CAN-ID des Objekts handeln, mit dem es auf dem CAN-Bus kommuniziert wird. Da eine Klasse mitunter mehrere *Data Properties* haben kann, kann diese Annotation als einzige mehrmals für eine Klasse auftreten.

Art der Datenänderung – die SourceType Annotation

Diese Annotation gibt dem Modellierer die Möglichkeit einzustellen, auf welche Art die Data Properties dieser Klasse zur Laufzeit mit realen Daten befüllt werden. Zur Auswahl stehen dazu die beiden Werte *Source_Push* und *Source_Pull*. Während *Source_Push* signalisiert, dass Daten von der Datenquelle aktiv in die Ontologie integriert werden, signalisiert *Source_Pull*, dass das kontextsensitive System die Daten manuell abfragen und in die Ontologie integrieren muss.

Häufigkeit der Datenänderung - die SourceFrequencyAnnotation

Wurde die Annotation *SourceType* mit dem Wert *Source_Pull* befüllt, kann der Modellierer durch diese Annotation festlegen, in welcher Frequenz das kontextsensitive System neue Werte der Data Properties bei den jeweiligen Datenquelle abfragt. In allen anderen Fällen hat diese Annotation keine Funktion.

Umgang mit redundanten Datenquellen - die SourceSelection Annotation

Mit dieser Annotation stehen dem Modellierer verschiedene Möglichkeiten zur Verfügung, die den Umgang mit mehreren Datenquellen definieren. Tabelle 4.6 fasst die möglichen Werte dieser Annotation und ihre Auswirkungen zusammen.

Umgang mit redundanten Daten - die ValueCombination Annotation

Wurde vom Modellierer die SourceSelection Annotation einer Klasse mit den Werten *Select_MergeAll* oder *Select_MergeSome* belegt, bietet ihm diese Annotation die Möglichkeit zu definieren, wie genau die multiplen Werte miteinander kombiniert werden sollen. Tabelle 4.7 fasst die möglichen Werte dieser Annotation zusammen. Die Auswirkungen dieser Annotation beziehen sich lediglich auf Data Properties eines numerischen Datentyps.

Wert der Annotation	Beschreibung
All	Alle Werte aus allen Quellen werden verwendet und redundant im System abgelegt und behandelt.
HighestConf	Aus allen Quellen wird der Wert mit dem höchsten Wahrheitswert für die weitere Verwendung ausgewählt. Alle anderen Werte werden verworfen.
HighesWeight	Aus allen Quellen wird der Wert mit dem höchsten Gewicht für die weitere Verwendung ausgewählt. Alle anderen Werte werden verworfen.
MergeAll	Alle Werte werden miteinander kombiniert verwendet, wenn jede angegebene Quelle aktuell die entsprechende Werte zur Verfügung stellen kann. Wie genau die Werte kombiniert werden, kann über die *ValueCombination* Annotation spezifiziert werden.
MergeSome	Alle Werte werden miteinander kombiniert verwendet, wenn mindestens eine der angegebenen Quellen aktuell die entsprechende Werte zur Verfügung stellen kann. Wie genau die Werte kombiniert werden, kann über die *ValueCombination* Annotation spezifiziert werden.

Tabelle 4.6: Mögliche Werte der SourceSelection Annotation

Wert der Annotation	Beschreibung
Average	Es wird der Durchschnitt aller Werte gebildet und als Grundlage für weitere Berechnungen verwendet.
WeightedAverage	Es wird ein gewichteter Durchschnitt gemäß des Gewichts aller Werte gebildet und als Grundlage für weitere Berechnungen verwendet.
ConfAverage	Es wird ein gewichteter Durchschnitt gemäß des Wahrheitswerte aller Werte gebildet und als Grundlage für weitere Berechnungen verwendet.
ConfWeightedAverage	Es wird ein gewichteter Durchschnitt gemäß des Gewichts und des Wahrheitswertes aller Werte gebildet und als Grundlage für weitere Berechnungen verwendet.

Tabelle 4.7: Mögliche Werte der ValueCombination Annotation

Wert der Annotation	Beschreibung
Product	Berechnung des Wahrheitswertes gemäß der Produktnorm.
MinMaxZadeh	Berechnung des Wahrheitswertes gemäß der Zadeh-Norm.
MinMaxLuka	Berechnung des Wahrheitswertes gemäß der Łukasiewicz-Norm.

Tabelle 4.8: Mögliche Werte der CalculationNorm Annotation

Berechnung von Wahrheitswerten - die CalculationNorm Annotation

Wie der Wahrheitswert höherwertiger Kontextinformationen berechnet werden kann, kann vom Modellierer mit Hilfe dieser Annotation eingestellt werden. Tabelle 4.8 fasst die möglichen Ausprägungen dieser Annotation sowie die dazugehörigen Implikationen zusammen.

4.3 Zusammenfassung

Im ersten Abschnitt dieses Kapitels wurde erläutert, welche Techniken zur Modellierung kontextsensitiver Systeme nach dem aktuellen Stand der Technik zur Verfügung stehen. Da bestehende Techniken zur Modellierung von Kontextinformationen bereits in verschiedenen Vergleichen gegeneinander bewertet wurden und die dabei angesetzten Vergleichskriterien eine sehr große Überdeckung mit den im Rahmen dieser Arbeit ermittelten Anforderungen aufweisen, wurden die Ergebnisse dieser Vergleiche als Grundlage für die Entscheidung herangezogen, Ontologiebasierte Modelle, ergänzt um Objektorientierte Ansätze, als am geeignetsten für die Modellierung kontextsensitiver FIS auszuwählen.

Im zweiten Abschnitt wurde gezeigt, wie Ontologiebasierte Modelle dazu genutzt werden können, die reale Welt mit ihren Daten, Fakten und Zusammenhängen zu erfassen. Dabei wurde detailliert darauf eingegangen, wie Data Properties, Subklassen, Datenbereiche und SWRL-Regeln genutzt werden können, um höherwertige Kontextinformationen im Sinne eines Situationsverständnisses aus einfachen Kontextinformationen abzuleiten. Im gleichen Abschnitt wurde abschließend auch gezeigt, wie Ontologiebasierte Modelle erweitert werden können, um Unsicherheiten und Redundanzen von Kontextinformationen behandeln zu können. Dazu wird vorgeschlagen, die aus der Fuzzy-Logik bekannten t-Normen und t-Conormen einzusetzen, um die Unsicherheiten von höherwertigen Kontextinformationen zu errechnen, die zuvor aus einfachen Kontextinformationen ermittelt wurden. Alle in diesem Kapitel vorgestellten Metainformationen modellierter Kontextinformationen werden über Annotationen den entsprechenden Klassen der Ontologie zugewiesen und können zur Laufzeit verwendet werden, um die Vorgaben des Modelliers einzuhalten. In Kapitel 6 werden diese Annotation erneut eine übergeordnete Rolle einnehmen, wenn gezeigt wird, wie aus dem Modell ein zur Laufzeit ein kontextsensitives FIS erstellt werden wird.

5. Modellierung von Nutzerintentionen

„One of my few shortcomings is that I can't predict the future."

(Lars Ulrich, Mitbegründer der Heavy Metal Band Metallica, [Ulr96])

Nutzerintentionen beschreiben, welche Absichten Nutzer bei der Bedienung eines Systems verfolgen. Sie beschreiben, welche Funktionen des Systems der Nutzer als nächstes mit welchem Ziel benutzen möchte. Zur Bestimmung der Nutzerintentionen ist die Situation, in der sich der Nutzer und das System aktuell befinden, ein entscheidender Faktor. Daher sind die modellierten Kontextinformationen, wie in Kapitel 4 beschrieben, die Basis für jede Nutzerintention.

In diesem Kapitel werden verschiedene Techniken und Methoden erläutert, mit deren Hilfe Nutzerintentionen, die von einem FIS erkannt werden sollen, modelliert werden können. Der Fokus liegt dabei auf der Modellierung von Nutzerintentionen und deren Wahrscheinlichkeit in Abhängigkeit zu für diese Nutzerintentionen relevanten Situationen. Jede dieser vorgestellten Techniken wird im Hinblick auf die in Kapitel 3 ermittelten Anforderungen bewertet, so dass am Ende dieses Kapitels eine Empfehlung für eine zu verwendende Technik zur Modellierung von Nutzerintentionen in FIS ausgesprochen werden kann.

Dieses Kapitel ist wie folgt aufgebaut: In Abschnitt 5.1 wird gezeigt, wie *Entscheidungsbäume* [SM64] zur Modellierung von Nutzerintentionen genutzt werden können und wie deren Verwendung im Rahmen dieser Arbeit bewertet wird. Anschließend wird in Abschnitt 5.2 präsentiert, wie und mit welchen Einschränkungen Nutzerintentionen für FIS mit Hilfe von *Künstlichen Neuronalen Netzen* [Roj96] modelliert werden können. Als letzte Technik zur Modellierung von Nutzerintentionen wird in Abschnitt 5.3 gezeigt, wie *Bayes'sche Netze* zur Modellierung von Nutzerintentionen verwendet werden können. Abschnitt 5.4 schlägt einen Prozess zur Modellierung von Nutzerintentionen vor, bevor Abschnitt 5.5 das Kapitel zusammenfasst.

Die Inhalte dieses Kapitels wurden bereits in [LSSS15] und [LSSS16] veröffentlicht.

Fortlaufendes Beispiel

Zur besseren Vergleichbarkeit der im Laufe dieses Kapitels vorgestellten Techniken zur Modellierung von Nutzerintentionen, wird ein fortlaufendes Beispiel eingeführt. Dieses Beispiel wird verwendet, um zu zeigen, wie das jeweilige Modell bei der Verwendung verschiedener Techniken zur Modellierung von Nutzerintentionen konkret gestaltet ist, und um zu demonstrieren, welche Vor- und Nachteile diese mit sich bringen. Für die Modellierung von Nutzerintentionen wird das bereits in Kapitel 4 eingeführte Beispiel zur Veranschaulichung der Modellierung von Kontextinformationen erweitert.

Die Erweiterung des Beispiels geht davon aus, dass das FIS über verschiedene Quellen zur Wiedergabe von Musik verfügt. Dazu zählen exemplarisch die Wiedergabe von Radio, die Wiedergabe von auf dem Mobiltelefons gespeicherter Musik und die Wiedergabe von Musik, die sich auf dem internen Speicher des FIS befindet. Für die Wiedergabequellen *Mobiltelefon* und *interner Speicher* wird davon ausgegangen, dass Informationen über das *Genre* der dort gespeicherten Musik verfügbar sind.

Für dieses Beispiel wird ferner angenommen, dass Nutzer eines FIS im Hinblick auf die Medienwiedergabe zwei Nutzerintentionen mit unterschiedlichen Ausprägungen haben können. Die erste Nutzerintention beschreibt, welche Quelle der Nutzer zur Wiedergabe von Musik verwendet möchte. Dabei stehen die Ausprägungen `Radio` zur Wiedergabe von Radio, `Mobile` zur Wiedergabe von Musik auf dem Mobiltelefon und `Internal` zur Wiedergabe von Musik des internen Speichers zur Verfügung. Diese Nutzerintention wird über den Namen `Source` identifiziert. Die zweite Nutzerintention beschreibt, welches Genre an Musik der Nutzer hören möchte. Um die Komplexität nicht unnötig zu erhöhen, beschränkt sich das Beispiel dafür auf die Ausprägungen `Rock` für Musik des Genre Rock, `Pop` für Musik des Genre Pop und `Jazz` für Musik des Genre Jazz. Der Name dieser zweiten Nutzerintention wird auf `Genre` festgelegt.

Eine letzte Annahme, die für dieses Beispiel getroffen wird, beschreibt, dass die beiden genannten Nutzerintentionen von den bereits in Kapitel 4 eingeführten Situationen `Drowsiness` und `AloneInCar` abhängen. Für die Nutzerintention `Source` wird exemplarisch davon ausgegangen, dass der Nutzer überwiegend Radio hört, wenn er nicht allein im Fahrzeug ist, während er zur Wiedergabe von auf dem Mobiltelefon gespeicherten Musik tendiert, wenn er sich allein im Fahrzeug befindet. Den internen Speicher seines FIS nutzt er nur gelegentlich und ohne erkennbare Muster. Für die Nutzerintention `Genre` geht dieses Beispiel davon aus, dass der Nutzer überwiegend Rock- und Popmusik hört, wenn er müde ist, jedoch zu Musik des Genre Jazz tendiert, wenn er wach ist. Ferner wird für diese Nutzerintention angenommen, dass er Rockmusik präferiert, wenn er allein im Fahrzeug ist, wohingegen er Popmusik hören möchte, wenn er nicht allein im Fahrzeug ist.

Abschließend sei angemerkt, dass dieses Beispiel keinen essentiellen Bestandteil dieser Arbeit darstellt, sondern lediglich zur Veranschaulichung der im weiteren Verlauf vorgestellten Techniken und Methoden zur Modellierung von Nutzerintentionen für FIS dient.

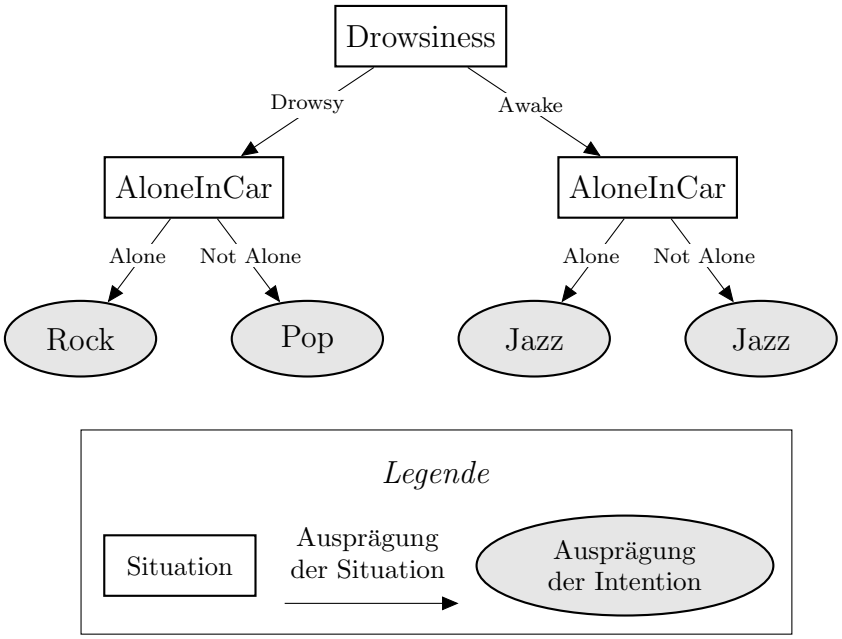

Abbildung 5.1: Modell der Nutzerintention `Genre` als Entscheidungsbaum

5.1 Modellierung durch Entscheidungsbäume

Entscheidungsbäume bieten die Möglichkeit, bestimmte Fragestellungen in Abhängigkeit zur Ausprägung gegebener Fakten, zu beantworten [SM64]. Im Hinblick auf die Modellierung von Nutzerintentionen können sie daher auch genutzt werden, um die Ausprägung einer Nutzerintention, in Abhängigkeit zur Ausprägung relevanter Situationen, zu bestimmen. Dabei bildet der Modellierer auf jeder Ebene des Entscheidungsbaumes die für eine Nutzerintention relevante Situation als Knoten des Baumes ab. Die Anzahl der Kindknoten dieses Knotens wird dabei durch die Anzahl der Ausprägungen der jeweiligen Situation bestimmt. In der darunterliegenden Ebene kann vom Modellierer dann die nächste relevante Situation als Knoten abgebildet werden. Nach diesem Schema entsteht Stück für Stück ein Entscheidungsbaum, dessen Blattknoten Ausprägungen der zu modellierenden Nutzerintention repräsentieren.

Der in Abbildung 5.1 dargestellte Entscheidungsbaum zeigt, wie auf diese Weise die Nutzerintention `Genre` modelliert werden kann. Auf der ersten Ebene des Entscheidungsbaumes wird die Situation `Drowsiness` abgebildet. Die beiden Ausprägungen dieser Situation (`Drowsy` und `Awake`) führen dann auf die zweite Ebene des Entscheidungsbaumes, die die Situation `AloneInCar` repräsentiert. Da `Drowsiness` zwei Ausprägungen hat, muss die Situation `AloneInCar` (ohne Anwendung von Optimierungen) zwei mal als Knoten im Entscheidungsbaum dargestellt werden. Da auch `AloneInCar` zwei Ausprägungen hat (`Alone` und `Not Alone`), führen von jedem Knoten, der `AloneInCar` repräsentiert, erneut zwei Kanten auf die nächste Ebene des Entscheidungsbaums, sodass sich in der dritten Ebene bereits vier Knoten ergeben. Da die Nutzerintention `Genre` nur von diesen beiden Situationen abhängt, besteht die dritte Ebene bereits aus den Blattknoten des Entscheidungsbaumes, die die verschiedenen Ausprägungen `Rock`, `Pop` und `Jazz` repräsentieren. Dem zu Beginn des Kapitels eingeführten Beispiels entsprechend sind diese Blattknoten mit den entsprechenden Ausprägungen gefüllt. So lässt sich beispielsweise erkennen, dass

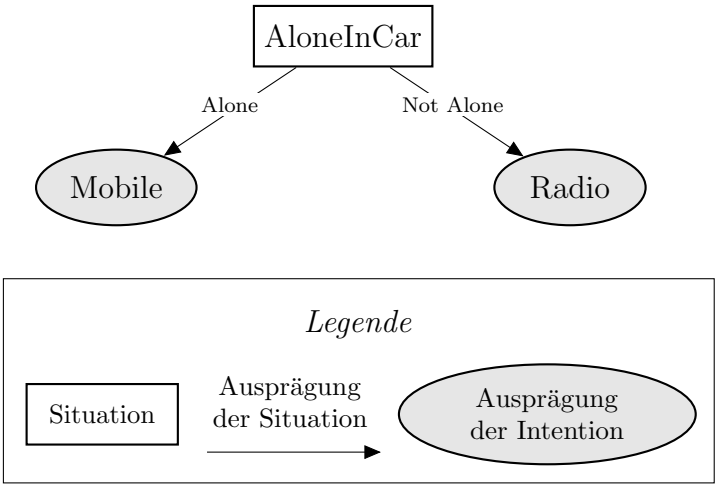

Abbildung 5.2: Modell der Nutzerintention `Source` als Entscheidungsbaum

der Nutzer zu Rock- und Popmusik tendiert, wenn er müde ist, während er Jazzmusik hören möchte, wenn er wach ist. Der Wunsch nach Jazzmusik wurde in diesem Beispiel vom Modellierer so stark interpretiert, dass die Situation `AloneInCar` keinerlei Einfluss auf die Nutzerintention `Genre` hat, wenn der Nutzer wach ist, da beide Blattknoten im rechten Teilbaum mit `Jazz` gefüllt wurden. Der Präferenz für Rockmusik in der Situation `Alone` wurde hingegen im linken Teilbaum durch den Modellierer Rechnung getragen.

Auf eine vergleichbare Art und Weise kann so auch die Nutzerintention `Source` modelliert werden. Das Ergebnis ist ein kleinerer Entscheidungsbaum, der in Abbildung 5.2 dargestellt ist.

Um zu überprüfen, welche Nutzerintention ein Nutzer in einer Situation hat, können die einzelnen Pfade des Entscheidungsbaumes als logische Regeln abgebildet und zur Laufzeit überprüft werden. Für den in Abbildung 5.1 gezeigten Entscheidungsbaum ergeben sich dabei die folgenden Regeln:

1. Drowsy $\wedge$ Alone $\Rightarrow$ Rock
2. Drowsy $\wedge$ Not Alone $\Rightarrow$ Pop
3. Awake $\wedge$ Alone $\Rightarrow$ Jazz
4. Awake $\wedge$ Not Alone $\Rightarrow$ Jazz

Dabei fällt auf, dass Regel 3 und 4 durch eine Optimierung durch eine einzige Regel vereinfacht und ersetzt werden können:

5. Awake $\Rightarrow$ Jazz

Auch im Entscheidungsbaum könnte die Überprüfung der Situation `AloneInCar` im rechten Teilbaum entfallen. Der durch diese Optimierung entstehende Entscheidungsbaum ist in Abbildung 5.3 dargestellt. Ist der Nutzer beispielsweise müde

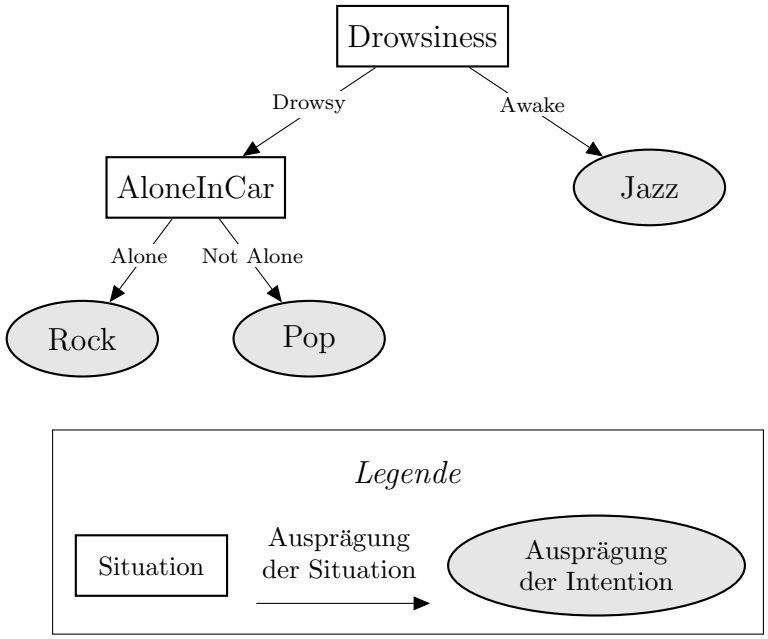

Abbildung 5.3: Modell der Nutzerintention `Genre` als vereinfachter Entscheidungsbaum

(`Drowsy`) und nicht allein im Fahrzeug (`Not Alone`), würde die Nutzerintention erkannt werden, dass der Fahrer Popmusik hören würde (`Pop`).

Bei genauerer Betrachtung des Beispiels fällt auf, dass für Radiowiedergabe keine Informationen über das Genre zur Verfügung stehe bzw. das System darauf keinen Einfluss nehmen kann. Ist der Nutzer zur Laufzeit des Systems müde und nicht allein im Fahrzeug, würde das System nach Auswertung der Entscheidungsbäume jedoch zu dem Ergebnis kommen, dass der Nutzer Radio und Popmusik hören möchte. Der dabei entstehende Konflikt muss bei der Verwendung von Entscheidungsbäumen entweder durch nachgelagerte Systemkomponenten behandelt werden, oder es muss eine Möglichkeit geschaffen werden, Entscheidungsbäume miteinander zu verknüpfen.

Bewertung

Im Folgenden wird die Modellierung von Nutzerintentionen durch Entscheidungsbäume detailliert anhand der in Kapitel 3 ermittelten Anforderungen bewertet.

Anforderung A14: Wahrscheinlichkeit der Nutzerintention

Entscheidungsbäume bieten in ihrer ursprünglichen Form keine Möglichkeit, die Wahrscheinlichkeit für die Ausprägung einer Nutzerintention zu modellieren. Auch die Fähigkeit, für alle Kombinationen an Ausprägungen der relevanten Situationen, eine Verteilung für die Wahrscheinlichkeiten der Ausprägungen der Nutzerintentionen anzugeben, fehlt den Entscheidungsbäumen. So wäre es zum Beispiels wünschenswert, wenn der Modellierer die Möglichkeit hat pro Blattknoten anzugeben, welche Wahrscheinlichkeiten die Ausprägungen `Rock`, `Pop` und `Jazz` an diesem Blattknoten haben. Entscheidungsbäume können diese Anforderung daher nicht zufriedenstellend erfüllen.

Anforderung A15: Mehrere Nutzerintentionen zu einem Zeitpunkt

Da unterschiedliche Nutzerintentionen durch voneinander getrennte Entscheidungsbäume modelliert werden, können ohne Weiteres mehrere Nutzerintentionen zur gleichen Zeit von einem System erkannt werden, das Entscheidungsbäume zur Laufzeit auswertet. Diese Anforderung wird daher von Entscheidungsbäumen vollständig erfüllt.

Anforderung A16: Nutzerintentionen sind situationsabhängig

Intentionen können bei der Verwendung von Entscheidungsbäumen, wie gezeigt, in Abhängigkeit zu Situationen und deren Ausprägungen modelliert werden. Entscheidungsbäume erfüllen diese Anforderung daher auch komplett.

Anforderung A17: Anpassung an den individuellen Nutzer

Entscheidungsbäume können nicht nur, wie gezeigt, durch Experten erstellt werden, sondern können auch auf Basis eines Datensatzes (dem sogenannten Trainingsdatensatz) *gelernt* werden. Die dabei entstehenden Ergebnisse sind jedoch oft für den Menschen nur schwer mehr verständlich und enthalten Abhängigkeiten, die zwar im Trainingsdatensatz enthalten sind, aber auf die zu modellierende Nutzerintention allgemein nicht zutreffen. Diese Anforderung wird daher im Rahmen dieser Arbeit lediglich als teilweise erfüllt betrachtet.

Anforderung A18: Gelernte Nutzerintentionen können vergessen werden

Da Entscheidungsbäume das Lernen von Nutzerintentionen nur eingeschränkt unterstützen, wird auch diese Anforderung als teilweise erfüllt betrachtet.

Anforderung A19: Hohe Verständlichkeit des Modells

Bei der Modellierung von Nutzerintentionen ist der Modellierer insbesondere darauf bedacht, die Abhängigkeiten zwischen Nutzerintentionen und Situationen zu berücksichtigen. In einem vollständigen Entscheidungsbaum kann er für jeden Blattknoten eindeutig erkennen, wie Situationen ausgeprägt sein müssen, um die Ausprägung der Nutzerintention zu erreichen, die durch diese Blattknoten repräsentiert wird. Diese Tatsache macht es für den Modellierer zum Einen eindeutig erkennbar, für welche Kombination an Situationsausprägungen er Ausprägungen von Nutzerintentionen modelliert, und sorgt zum Anderen für eine intuitive Lesbarkeit des Modells. Diese intuitive Verständlichkeit hat jedoch einen hohen Preis: Entscheidungsbäume können schnell komplex werden und mitunter eine Vielzahl an Redundanzen enthalten. Hätte `Drowsiness` beispielsweise nicht nur zwei, sondern mehr Ausprägungen, wäre der Entscheidungsbaum womöglich schon auf zweiter Ebene zu breit und unübersichtlich geworden. Hängt eine Nutzerintention von mehr als zwei verschiedenen Situationen ab, bekommt der Entscheidungsbaum zusätzlich mehrere Ebenen, wodurch seine Komplexität erneut gesteigert wird. Das Modell der Nutzerintentionen kann dadurch sehr groß und unübersichtlich werden, was sich negativ auf die allgemeine Verständlichkeit des Modells auswirkt. Diese Anforderung wird daher als teilweise erfüllt betrachtet.

Anforderung A20: Modell ist standardisiert

Entscheidungsbäume sind im Bereich des *Data Minings* ein weit verbreiteter Ansatz [AW97]. Beim Einsatz von Entscheidungsbäumen zur Modellierung von Nutzerintentionen müssen daher keine proprietären Modelle oder Formate entwickelt werden. Diese Anforderung wird daher als erfüllt betrachtet.

Anforderung A21: Vorhandensein einer Tool-Unterstützung

Aufgrund der hohen Verbreitung und Standardisierung von Entscheidungsbäumen, gibt es auch zahlreiche Software-Tools, die bei der Erstellung von Entscheidungsbäumen herangezogen werden können. Dazu zählen zum Beispiel Tools für *R*[1] oder IBMs *SPSS*[2].

Zusammenfassung

Dieser Abschnitt hat gezeigt, dass die Verwendung von Entscheidungsbäumen zur Modellierung von Nutzerintentionen zwar möglich ist, im Hinblick auf die ermittelten Anforderungen jedoch lediglich unzureichend zufriedenstellend wäre, da nur vier der acht Anforderungen als vollständig erfüllt betrachtet werden können.

5.2 Modellierung durch Neuronale Netze

Nutzerintentionen können auch durch ein KNN [Roj96] modelliert werden. Eine konkrete Möglichkeit der Modellierung besteht dabei darin, die für die jeweilige Nutzerintention relevanten Situationen und deren Ausprägungen als *Eingangsneuronen* und die einzelnen Ausprägungen der zu erkennenden Nutzerintention als *Ausgangsneuronen* des Netzes abzubilden. Anschließend können mit Hilfe einer vollständigen Vernetzung (jedes Ausgangsneuron ist mit jedem Eingangsneuron verknüpft), Kantengewichten und den Schwellwerten aller Neuronen, die Abhängigkeiten zwischen Situationen und Nutzerintentionen modelliert werden. Bei der grafischen Darstellung eines solchen KNNs können diejenigen Kanten entfallen, deren Kantengewicht auf 0 gesetzt wurde, da eine solche Kante keinerlei Auswirkung auf das Neuron am Ende dieser Kante hätte und die Darstellung dadurch übersichtlicher wird.

Abbildung 5.4 zeigt, wie die Nutzerintentionen `Genre` und `Source` als KNN modelliert werden können. Auf der linken Seite des Netzes sind die Ausprägungen der für die Intentionen relevanten Situationen als Eingangsneuronen dargestellt (`Awake`, `Drowsy`, `Alone`, `Not Alone`). Die restlichen Neuronen repräsentieren Ausprägungen der Nutzerintentionen `Genre` und `Source`. Für die Ausprägung `Radio` der Intention `Source` wurde entsprechend den Vorgaben des Beispiels, eine Abhängigkeit zwischen den Neuronen `Not Alone` und `Radio` modelliert. Diese Abhängigkeit manifestiert sich im KNN als Kante zwischen den Neuronen `Not Alone` und `Radio`. Im Gegensatz zu Entscheidungsbäumen bieten KNNs auch die

[1] The R Project for Statistical Computing.
Website: https://www.r-project.org/. Zuletzt überprüft am 19.10.2016.

[2] Statistical Package for the Social Sciences.
Website: http://www.ibm.com/software/de/analytics/spss/. Zuletzt überprüft am 19.10.2016.

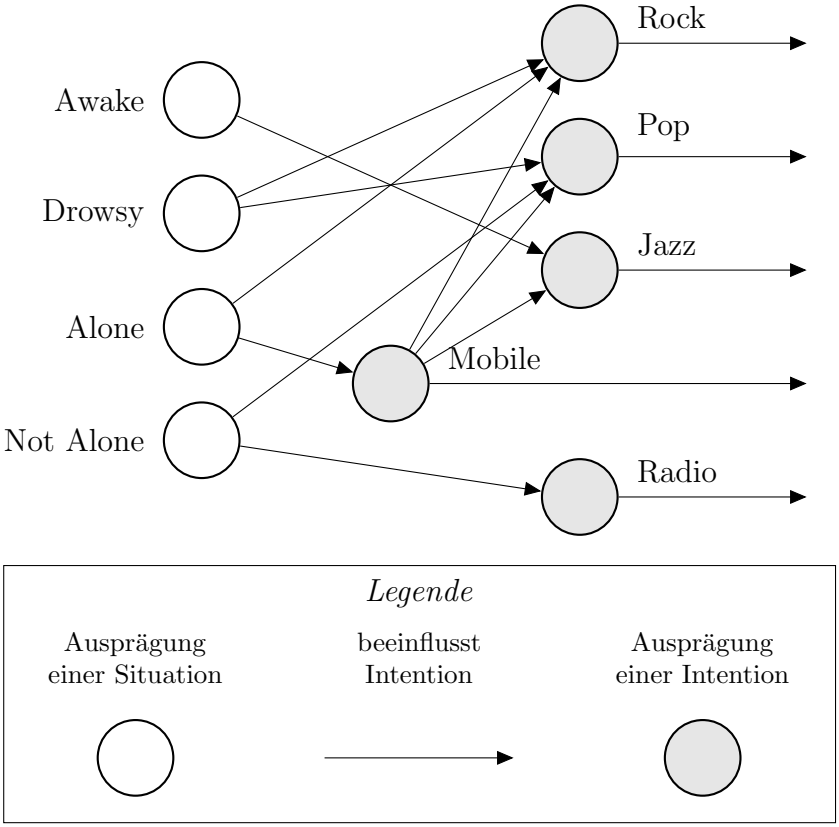

Abbildung 5.4: Modell der Nutzerintentionen `Genre` und `Source` als KNN

Möglichkeit, Abhängigkeiten zwischen Nutzerintentionen im Modell abbilden zu können. Da die Nutzerintention `Genre` nur dann relevant ist, wenn zeitgleich auch die Nutzerintention `Mobile` erkannt wurde, wurde im KNN eine Abhängigkeit zwischen der Ausprägung `Mobile` und den Ausprägungen `Rock`, `Pop` und `Jazz` modelliert.

Zur Laufzeit könnten Eingangsneuronen mit 0 befüllt werden, wenn die Situationsausprägung, die sie repräsentieren, zur Zeit nicht gegeben ist, oder mit 1 befüllt werden, wenn sie gegeben ist. Der Schwellwert der Eingangsneuronen könnte auf 0.5 gesetzt werden, um sicherzustellen, dass Eingansneuronen nur dann *feuern*, wenn ihr Wert 1 beträgt. Kanten zwischen Eingangs- und Ausgangsneuronen könnte ein Gewicht von 1 erhalten, wenn die Verbindung, die sie repräsentieren, relevant ist und andernfalls 0. Der Schwellwert der Ausgangsneuronen könnte vom Modellierer nun verwendet werden, um zu steuern, wann die Nutzerintentionsausprägung, die sie repräsentieren, gelten soll. Im einfachsten Fall wird der Schwellwert gleich der Summe der relevanten Kanten gesetzt, sodass alle Eingangsneuronen feuern müssen, um den Schwellwert des Ausgangsneuronen zu überschreiten (unter der Bedingung, dass die Kantengewichte wie beschrieben auf 1 oder 0 gesetzt wurden).

Eine Nutzerintention liegt dann vor, wenn ein Ausgangsneuron des KNNs feuert, also wenn seine Eingaben seinen Schwellwert überschreiten. Dieses Feuern kann als binäre Operation verstanden werden: entweder das Neuron feuert, oder es feuert nicht. Mit Bezug auf die Nutzerintention, die es repräsentiert, bedeutet das: entweder der Nutzer hat diese Intention, oder er hat sie nicht. Werden KNNs jedoch mit mehreren Ebenen abgebildet, so können Zwischenebenen dazu genutzt werden, um

den Wahrheitswert des Auslösens eines Neurons zu beschreiben. Solche KNNs sind auch als *Probabilistische Neuronale Netze* bekannt [Spe88, Spe90]. Die Komplexität solcher Netze ist jedoch deutlich gesteigert, sodass sie nicht manuell erstellt werden können, sondern auf Basis eines Datensatzes gelernt werden müssen. Bei einem solchen Training kommt es mitunter zu Problemen und Netzen, die aktuell noch nicht vollständig verstanden werden können [GB10]. Auch werden KNNs, neben anderen Technologien, aktuell im Forschungsgebiet *Deep Learning* von Wissenschaft und Industrie untersucht, weiterentwickelt und für eine Vielzahl an Szenarien eingesetzt. Eine Übersicht zu aktuellen Aktivitäten und Herausforderungen bei der Verwendung von KNNs lässt sich in [Sch15] finden.

Bewertung

Im Folgenden wird die Modellierung von Nutzerintentionen durch KNNs detailliert anhand der in Kapitel 3 ermittelten Anforderungen bewertet.

Anforderung A14: Wahrscheinlichkeit der Nutzerintention

Einfache, manuell erstellte KNNs bieten keine Möglichkeit, Wahrscheinlichkeiten für Nutzerintentionen zur Laufzeit zur Verfügung zu stellen. Erst durch das Hinzufügen weiterer Schichten, kann diese Anforderung erfüllt werden. Allerdings steigt dabei die Anzahl der Neuronen, der Grad ihrer Vernetzung und damit auch die Verständlichkeit des Netzes insgesamt, weshalb diese Anforderung nur als teilweise erfüllt betrachtet wird.

Anforderung A15: Mehrere Nutzerintentionen zu einem Zeitpunkt

Wie in Abbildung 5.4 gezeigt, können KNNs dazu genutzt werden mehrere Nutzerintentionen in einem Netz abzubilden. Im Gegensatz zur Verwendung von Entscheidungsbäumen hilft diese Tatsache dabei, Abhängigkeiten zwischen Nutzerintentionen besser modellieren zu können. Neuronen, die Nutzerintentionen repräsentieren, können auch gleichzeitig feuern, wodurch es möglich wird, das mehrere Nutzerintentionen zeitgleich vom KNN erkannt werden können. Diese Anforderung wird daher als vollständig erfüllt betrachtet.

Anforderung A16: Nutzerintentionen sind situationsabhängig

Intentionen können auch bei der Verwendung von KNNs situationsabhängig modelliert werden. Dazu werden, wie gezeigt, Neuronen verwendet, die Situationsausprägungen repräsentieren. Je nach Anzahl relevanter Situationen und deren Ausprägungen kann die Zahl der sie repräsentierenden Neuronen rapide ansteigen. Nichtsdestotrotz ist diese Anforderung als erfüllt zu betrachten.

Anforderung A17: Anpassung an den individuellen Nutzer

Der vorgestellte Prozess geht von einer manuellen Modellierung der Nutzerintention durch einen Experten aus. Genau wie Entscheidungsbäume können jedoch auch KNNs, basierend auf einem Trainingsdatensatz, gelernt werden [Roj96]. Dabei werden Kantengewichte und Schwellwerte iterativ so lange angepasst, bis eine möglichst gute Vorhersage der Trainingsdaten erreicht wird. Als Trainingsdatensatz können auch aufgezeichnete Nutzerinteraktionen dienen, auf deren Basis das KNN an die individuellen Bedürfnissen der Nutzer adaptiert werden kann. Diese Anforderung wird daher als vollständig erfüllt betrachtet.

Anforderung A18: Gelernte Nutzerintentionen können vergessen werden

So wie KNNs an den individuellen Nutzer angepasst werden können (vgl. Bewertung Anforderung A17), können gelernte Nutzerintentionen auch wieder vergessen werden, wenn sie im Trainingsdatensatz nicht mehr vorkommen. Diese Anforderung wird daher ebenfalls als vollständig erfüllt betrachtet.

Anforderung A19: Hohe Verständlichkeit des Modells

Wie bereits erwähnt, können KNNs manuell oder automatisch erstellt bzw. gelernt werden. Bei der manuellen Erstellung muss der Modellierer alle Neuronen, Kanten, Kantengewichte und Schwellwerte festlegen. Für das Festlegen von Kantengewichten und Schwellwerten ist ein Verständnis über die Funktionsweise von KNNs von Nöten. Dieses Grundwissen kann je nach Ausbildung des Modellierers jedoch nicht vorausgesetzt werden. Werden KNNs automatisch gelernt besteht die Gefahr, dass das Netz für den Menschen nicht mehr nachvollziehbar ist. Die Verständlichkeit von KNNs zur Modellierung von Nutzerintention wird im Rahmen dieser Arbeit daher als gering eingestuft und diese Anforderung daher als nicht erfüllt betrachtet.

Anforderung A20: Modell ist standardisiert

Wie oben bereits erwähnt, finde KNNs aktuell vielseitigste Anwendungen im Bereich des *Deep Learning* [Sch15], weshalb das Modell an sich als standardisiert betrachtet werden kann. Sie erfüllen daher diese Anforderung.

Anforderung A21: Vorhandensein einer Tool-Unterstützung

Aufgrund der gestiegenen Popularität von KNNs ist auch eine Vielzahl an Software-Tools für deren Entwicklung verfügbar. Insbesondere das von Google entwickelte und als Open Source Projekt veröffentlichte *Tensorflow*[3] erlangt im Bereich der Künstlichen Intelligenz eine immer höhere Beliebtheit. Auch für *MATLAB*[4] und das bereits erwähnte Tool *R*[5] sind Erweiterungen zum Umgang mit KNNs verfügbar. Diese Anforderung wird daher ebenfalls als erfüllt betrachtet.

Zusammenfassung

Zusammenfassend lässt sich sagen, dass sechs der acht ermittelten Anforderungen beim Einsatz von KNNs zur Modellierung von Nutzerintentionen als vollständig erfüllt betrachtet werden können. Sie bieten daher eine solide Grundlage um Nutzerintentionen zu modellieren. Während Anforderung A14 durch den Einsatz Probabilistischer Neuronaler Netz auch vollständig erfüllt werden kann, sorgt der geringe Grad an Verständlichkeit dafür, dass KNNs für eine manuelle oder semi-automatische Modellierung von Nutzerintentionen insgesamt im Rahmen dieser Arbeit als nicht ausreichend eingestuft werden. Aus diesem Grund wird im nächsten Abschnitt untersucht, ob sich eine besser geeignete Technik zur Modellierung von Nutzerintentionen finden lässt.

[3] Tensorflow Website: https://www.tensorflow.org/. Zuletzt überprüft am 19.10.2016.
[4] Erweiterung für MATLAB: *Neural Network Toolbox*.
Website: http://de.mathworks.com/products/neural-network/. Zuletzt überprüft am 19.10.2016.
[5] Erweiterung für R: *neuralnet*.
Website: https://cran.r-project.org/web/packages/neuralnet/. Zuletzt überprüft am 19.10.2016.

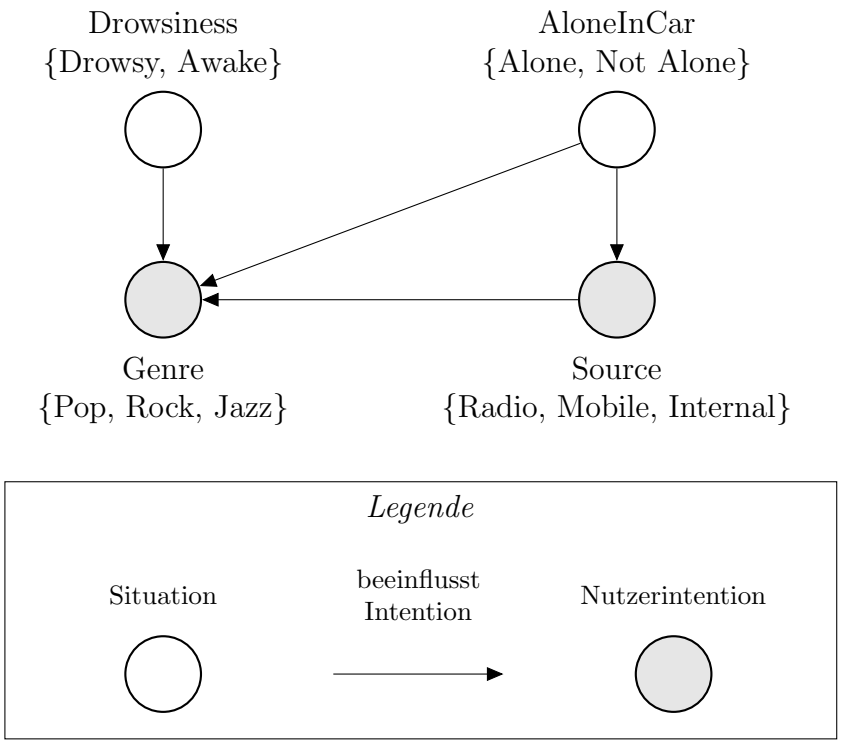

Abbildung 5.5: Modell der Nutzerintention `Genre` und `Source` als Bayes'sches Netz

5.3 Modellierung durch Bayes'sche Netze

Die Untersuchungen in Abschnitt 5.2 haben gezeigt, dass der größte Nachteil bei der Verwendung von KNNs zur Modellierung von Nutzerintentionen die Verständlichkeit des Modells ist. Da Bayes'sche Netze [BP63, Pea85] zunächst lediglich bedingte Wahrscheinlichkeiten visualisieren, kann davon ausgegangen werden, dass sie besser verstanden werden können als mehrschichtige KNNs, die – beim Einsatz Probabilistischer Neuronaler Netze – Neuronen enthalten, die weder Situationen noch Nutzerintentionen entsprechen. Basierend auf dieser Annahme wird in diesem Abschnitt zunächst zunächst gezeigt, wie die Elemente Bayes'scher Netze verwendet werden können, um Nutzerintentionen zu modellieren. Anschließend wird, erneut anhand der Anforderungen aus Kapitel 3, die Eignung Bayes'scher Netze für eben diese Aufgabe bewertet.

Eine Möglichkeit zur Modellierung von Nutzerintentionen mit Hilfe Bayes'scher Netze besteht darin, die zu modellierenden Intentionen und die Situationen, von denen die Intentionen abhängen, als Knoten des Bayes'schen Netzes abzubilden. Die Abhängigkeiten zwischen Situationen und Nutzerintentionen werden durch gerichtete Kanten abgebildet. Abbildung 5.5 zeigt die Nutzerintentionen `Genre` und `Source` als Bayes'sches Netz.

Für die Knoten, die die Nutzerintentionen `Genre` und `Source` repräsentieren, müssen anschließend zusätzlich die dazugehörigen Wahrscheinlichkeitstabellen erstellt werden. In Tabelle 5.1 ist die Wahrscheinlichkeitsverteilung für den Knoten `Source` aufgetragen. Sie zeigt, dass dem Beispiel entsprechend, modelliert wurde, dass die Nutzerintention `Mobile` die größte Wahrscheinlichkeit hat, wenn der Nutzer allein im Fahrzeug ist und dass die Nutzerintention `Radio` die größte Wahrscheinlichkeit

Ausprägung AloneInCar	Radio	Mobile	Internal
Alone	$^2/_{10}$	$^7/_{10}$	$^1/_{10}$
Not Alone	$^5/_{10}$	$^4/_{10}$	$^1/_{10}$

Tabelle 5.1: Wahrscheinlichkeitsverteilung für die von der Situation AloneInCar abhängende Nutzerintention Source (ergänzend zu Abbildung 5.5)

hat, wenn der Nutzer nicht allein im Fahrzeug ist. Analog dazu ist in Tabelle 5.2 die Wahrscheinlichkeitsverteilung für den Knoten Genre dargestellt. Da dieser Knoten von den Knoten Drowsiness, AloneInCar und Genre abhängt, ist für jede Ausprägungskombinationen dieser drei Knoten eine Zeile in der Wahrscheinlichkeitstabelle vorhanden. Die Tabelle zeigt, dass für die Ausprägungen Pop, Rock und Jazz eine Gleichverteilung modelliert wurde, wenn die Nutzerintention Source die Ausprägung Radio annimmt, da für das Radio als Wiedergabequelle dem Beispiel entsprechend ohnehin kein bestimmtes Genre gespielt werden kann (vgl. Zeile 1–4 der Tabelle 5.2). Die restlichen Zeilen der Tabelle zeigen, dass die Wahrscheinlichkeiten für die verschiedenen Ausprägungen der Nutzerintention Genre entsprechend des eingangs erwähnten Beispiels modelliert wurden. So definierte das Beispiel, dass der Nutzer gern Jazzmusik hört, wenn er allein im Fahrzeug ist. Die Tatsache lässt sich anhand der Zeilen 7, 8, 11 und 12 nachvollziehen.

Die Knoten, die Situationen repräsentieren, erhalten zur Laufzeit die der Situation entsprechenden Werte, wie zum Beispiel Drowsy und Alone. Für die Auswertung des Bayes'schen Netzen sind die Knoten von Bedeutung, die Intentionen widerspiegeln, wie zum Beispiel Genre und Source. Aufgrund der durch bedingte Wahrscheinlichkeiten modellierten Abhängigkeiten, sowie den dazugehörigen Wahrscheinlichkeitstabellen, kann in jeder beliebigen Situation die Wahrscheinlichkeitsverteilung für die modellierten Nutzerintentionen angegeben werden. Dabei kann das Bayes'sche Netz nicht nur die wahrscheinlichste Ausprägung einer Nutzerintention bestimmen, sondern ist in der Lage für jede Ausprägung, wie zum Beispiel Pop, Rock und Jazz, die Wahrscheinlichkeit in Abhängigkeit zur aktuellen Situation anzugeben.

5.3.1 Trainieren des Modells

Bei der Verwendung Bayes'scher Netze als Modell der Intentionen von Nutzer eines FIS, können verschiedene Teile des Modells mit Hilfe bereits bekannter Verfahren [CH92, SV95, Bou95, FGG97] automatisch erstellt werden. Der Aufwand zur Erstellung des Modells durch den Modellierer kann dadurch reduziert werden. Die zwei wesentlichsten Möglichkeiten zur automatischen Erstellung des Modells sind: das Trainieren der Struktur des Bayes'schen Netzes, sowie das Trainieren der Wahrscheinlichkeitsverteilungen. Beide Möglichkeiten sind unabhängig voneinander und können auch einzeln angewandt werden.

Beiden Verfahren basieren dabei auf Datensätzen, die durch Beobachtungen oder Aufzeichnungen (zum Beispiel im Rahmen von Probandenstudien) entstanden sind. Attribute dieser Datensätze sind sämtliche Situationen, die zur Laufzeit durch Kontextinformationen ermittelt wurden (mit Bezug auf das fortlaufende Beispiel die

Ausprägung `Source`	Ausprägung `Drowsiness`	Ausprägung `AloneInCar`	`Pop`	`Rock`	`Jazz`
Radio	Drowsy	Alone	1/3	1/3	1/3
Radio	Drowsy	Not Alone	1/3	1/3	1/3
Radio	Awake	Alone	1/3	1/3	1/3
Radio	Awake	Not Alone	1/3	1/3	1/3
Mobile	Drowsy	Alone	2/10	7/10	1/10
Mobile	Drowsy	Not Alone	5/10	4/10	1/10
Mobile	Awake	Alone	2/10	3/10	5/10
Mobile	Awake	Not Alone	1/10	4/10	5/10
Internal	Drowsy	Alone	2/10	7/10	1/10
Internal	Drowsy	Not Alone	5/10	4/10	1/10
Internal	Awake	Alone	2/10	3/10	5/10
Internal	Awake	Not Alone	1/10	4/10	5/10

Tabelle 5.2: Wahrscheinlichkeitsverteilung für die von den Situationen `AloneInCar` und `Drowsiness` und der Nutzerintention `Source` abhängende Nutzerintention `Genre` (ergänzend zu Abbildung 5.5)

Drowsiness	AloneInCar	Genre	Source
Drowsy	Alone	Rock	Mobile
Drowsy	Not Alone	Pop	Mobile
Drowsy	Not Alone	Pop	Radio
Drowsy	Not Alone	Pop	Mobile
Awake	Alone	Jazz	Mobile
Awake	Not Alone	Jazz	Mobile
Awake	Alone	Jazz	Radio
Awake	Not Alone	Jazz	Radio

Tabelle 5.3: Exemplarischer Trainingsdatensatz zum Lernen Bayes'scher Netze

Situationen `Drowsiness` und `AloneInCar`), sowie Informationen darüber, welche Intention der Nutzer des Systems zu diesem Zeitpunkt hatte. Die Erfassung der letztgenannten Information kann dabei je nach Intention komplex und aufwendig sein. Mit Blick auf das fortlaufende Beispiel könnte diese Information aber leicht über die im System eingestellte Wiedergabequelle (Intention `Source`) und das Genre des aktuelle Musiktitels (Intention `Genre`) erledigt werden. Ein so entstehender Datensatz ist exemplarisch in Tabelle 5.3 dargestellt. Da er zum *Trainieren* des Bayes'schen Netzes verwendet wird, wird er auch *Trainingsdatensatz* genannt.

5.3.1.1 Trainieren der Struktur

Die erste Möglichkeit zur Minimierung des Aufwandes für den Modellierer besteht darin, die Struktur der Bayes'schen Netzes auf Basis des Trainingsdatensatzes zu ermitteln – zu *trainieren*. Die Knoten des Bayes'schen Netzes werden dabei aus den gegebenen Attributen des Datensatzes abgeleitet. Basierend auf den Trainingsdatensätzen können verschiedene Algorithmen, wie zum Beispiel *K2*[CH92], *CB*[SV95], *Simulated Annealing*[Bou95], oder *TAN*[FGG97] eingesetzt werden, um Knoten und Kanten des Bayes'schen Netzes zu erstellen. Welcher dieser Algorithmen sich am besten zum Trainieren von Strukturen Bayes'scher Netze zum Zwecke der Modellierung von Nutzerintentionen eignet, hängt stark vom individuellen Anwendungsfall ab und kann im Rahmen dieser Arbeit daher nicht allgemeingültig bewertet werden.

Entscheidend für diese Art des Lernens ist der Trainingsdatensatz. Enthält dieser nicht alle relevanten Abhängigkeiten zwischen Situationen und Intentionen (zum Beispiel weil bei der Erstellung des Datensatzes nicht alle Kombinationen von Situationen aufgezeichnet wurden sind), können die Algorithmen diese nicht erkennen und entsprechend auch nicht in der Struktur des Bayes'schen Netzes abbilden. In den aufgezeichneten Daten des Trainingsdatensatzes können sich zusätzlich dazu auch Abhängigkeiten befinden, die nur aufgrund der gewählten Rahmenbedingungen der Aufzeichnung entstanden sind. Diese Tatsache führt unter Umständen dazu, dass sich im Bayes'schen Netz Kanten zwischen Knoten abbilden, die von Experten während einer manueller Modellierung nicht erstellt worden wären. Es empfiehlt sich daher, die trainierte Struktur des Bayes'schen Netzes von einem Experten überprüfen und

gegebenenfalls korrigieren zu lassen. Da die Struktur nur bedingte Wahrscheinlichkeiten abbildet, können solche Korrekturen mit entsprechender Software-Unterstützung schnell und leicht verständlich durchgeführt werden.

5.3.1.2 Trainieren der Wahrscheinlichkeitsverteilungen

Eine zweite Möglichkeit zum Trainieren des Modells ist das Trainieren der Wahrscheinlichkeitstabellen. Dazu kommen sogenannte *Schätzer* zum Einsatz, die basierend auf den Häufigkeiten der beobachteten Kombinationen an Situationen und Intentionen schätzen, welche bedingten Wahrscheinlichkeiten sich daraus für die Knoten des Bayes'schen Netzes ergeben. Die Struktur des Bayes'schen Netzes muss dabei bekannt sein. Ob sie jedoch manuell durch einen Experten erstellt wurde, oder ebenfalls durch ein Training wie in Abschnitt 5.3.1.1 beschrieben, erstellt wurde, ist dafür irrelevant. Ein weit weitverbreitetes Verfahren zum Schätzen der Wahrscheinlichkeitstabellen ist das BMA (vgl. Abschnitt 2.3.3). Die Möglichkeit, die Wahrscheinlichkeitstabellen zu trainieren, bringt zwei entscheidende Vorteile mit sich:

a) Der Modellierer wird bei der Modellierung von Nutzerintentionen stark entlastet. Insbesondere bei der Modellierung von Nutzerintentionen, die von einer Vielzahl an Situationen mit vielen Ausprägungen abhängen, können die Wahrscheinlichkeitstabellen Ausmaße annehmen, die für den Modellierer schwer zu überblicken sind und nur sehr aufwendig mit Inhalt gefüllt werden können. Das automatische Trainieren kann diese Aufgabe übernehmen, ohne dem Modellierer dabei die Möglichkeit zu versagen, die trainierten Wahrscheinlichkeiten manuell anzupassen.

b) Das automatische Trainieren der Wahrscheinlichkeitstabellen kann zur Laufzeit des Systems genutzt werden, um die Intentionserkennung an die individuellen Bedürfnisse der Nutzer anzupassen. Während der eine Nutzer in bestimmten Situationen zu Rockmusik tendiert, tendiert ein anderer zu Popmusik. Durch eine entsprechende Aufzeichnung des Nutzerverhaltens zur Laufzeit und einem wiederholten und fortlaufenden Training der Wahrscheinlichkeitstabellen, können sich individuelle Nutzerintentionen auch im Modell wiederfinden.

Bewertung

Im Folgenden wird die Modellierung von Nutzerintentionen durch Bayes'sche Netze detailliert anhand der in Kapitel 3 ermittelten Anforderungen bewertet.

Anforderung A14: Wahrscheinlichkeit der Nutzerintention

Wie bereits am fortlaufenden Beispiel gezeigt, sind Bayes'sche Netze in der Lage, Wahrscheinlichkeiten für Nutzerintentionen anzugeben. Dabei können sie in jeder Situation für jede Intention eine Wahrscheinlichkeitsverteilung für die einzelnen Ausprägungen dieser Situation angeben. Diese Anforderung wird daher als vollständig erfüllt betrachtet.

Anforderung A15: Mehrere Nutzerintentionen zu einem Zeitpunkt

Verschiedenen Nutzerintentionen, wie zum Beispiel `Genre` und `Source`, können, bei der Modellierung durch Bayes'sche Netze, zur Laufzeit zu einem Zeitpunkt erkannt werden. Auch Abhängigkeiten zwischen Intentionen können bei der Verwendung Bayes'scher Netze abgebildet werden. So hat die Intention `Source` beispielsweise eine Einfluss auf die Intention `Genre`, da nicht für alle Wiedergabequellen ein Genre vom System bestimmt werden kann. Diese Anforderung wird daher als vollständig erfüllt betrachtet.

Anforderung A16: Nutzerintentionen sind situationsabhängig

Durch das Hinzufügen von gerichteten Kanten und Wahrscheinlichkeitstabellen, wird es dem Modellierer ermöglicht, Nutzerintentionen und deren Ausprägungen in Abhängigkeit zu bestimmten Situationen und deren Ausprägungen zu setzen. Diese Anforderung wird daher als vollständig erfüllt betrachtet.

Anforderung A17: Anpassung an den individuellen Nutzer

Abschnitt 5.3.1 hat gezeigt, dass sowohl die Struktur des Bayes'schen Netzes, als auch die Wahrscheinlichkeitsverteilungen der einzelnen Knoten gelernt werden können. Diese Tatsache erlaubt es, das Bayes'sche Netz, und damit die Erkennung der Nutzerintentionen, zur Laufzeit adaptiert werden können. Wie gezeigt, müssen dazu die Nutzerintentionen und die aktuellen Situationen aufgezeichnet werden. Werden die vom Nutzer getätigten Interaktionen mit dem FIS dazu genutzt, um diese Aufzeichnungen durchzuführen, können sich Bayes'sche Netze auch an das individuelle Verhalten der Nutzer anpassen. Diese Anforderung wird daher als vollständig erfüllt betrachtet.

Anforderung A18: Gelernte Nutzerintentionen können vergessen werden

Durch das Adaptieren an den individuellen Nutzer des FIS, ist es auch möglich, zuvor gelernte Nutzerintentionen wieder zu vergessen. Dazu werden *alte* Aufzeichnungen aus der Liste aller Aufzeichnungen entfernt, wodurch sie beim Lernen des Bayes'schen Netzes keine weitere Berücksichtigung finden. Diese Anforderung wird daher als vollständig erfüllt betrachtet.

Anforderung A19: Hohe Verständlichkeit des Modells

Da alle Knoten im Bayes'schen Netz entweder Situationen oder Nutzerintentionen darstellen und deren Ausprägungen die Werte dieser Knoten ausmachen, ist für den Modellierer sofort ersichtlich, welche Bedeutung die einzelnen Knoten im Bayes'schen Netz haben. Die Abhängigkeiten zwischen Situationen und Nutzerintentionen (oder auch zwischen zwei Nutzerintentionen), sind durch ihre visuelle Darstellung als gerichtete Kanten leicht verständlich. Auch wenn die Wahrscheinlichkeitstabellen mitunter nur sehr aufwendig manuell zu erstellen sind, so ist ihre Verständlichkeit dennoch als hoch einzustufen, da sie immer nach dem gleichen Schema die Wahrscheinlichkeitsverteilung einer Intention in einer bestimmten Situation angeben. Diese Anforderung wird daher als vollständig erfüllt betrachtet.

Anforderung A20: Modell ist standardisiert

Die theoretischen Konzepte hinter Bayes'schen Netzen gehen bis in das hat 1763 zurück [BP63]. Seit den 1980er Jahren werden zum Abbilden bedingter Wahrscheinlichkeiten in Netzstrukturen verwendet [Pea85]. Mittlerweile existieren auch verschiedenen Dateiformate, die es erlauben Bayes'sche Netze zu persistieren. Die weit verbreitetsten Formate sind *BNIF*[6] bzw. dessen Nachfolger *XMLBIF*[7] Bayes'sche Netze werden daher als standardisiert und diese Anforderung demnach als vollständig erfüllt betrachtet.

Anforderung A21: Vorhandensein einer Tool-Unterstützung

Tools, wie beispielsweise *Java Bayes*[8] oder *Weka*[9], erlauben es Bayes'sche Netze zu erstellen, zu bearbeiten und automatisiert anhand von aufgezeichneten Datensätzen zu erlernen. Auch diese Anforderung wird daher als vollständig erfüllt betrachtet.

Zusammenfassung

Dieser Abschnitt hat gezeigt, wie Bayes'sche Netze dazu verwendet werden können, um Nutzerintentionen in Abhängigkeit von Situationen und deren Ausprägungen zu modellieren. Es wurde auch gezeigt, wie der Aufwand zum Erstellen der Netze und der Wahrscheinlichkeitstabellen durch automatisiertes Lernen reduziert werden kann. Zusammenfassend lässt sich zum Einsatz Bayes'scher Netzer zur Modellierung von Nutzerintentionen für FIS festhalten, dass alle in Kapitel 3 ermittelten Anforderungen durch Bayes'sche Netze erfüllt werden.

Fazit des Vergleichs

Tabelle 5.4 fasst den Vergleich der vorgestellten Techniken und Methoden abschließend zusammen. Entscheidungsbäume und KNNs haben vor allem die Nachteile, dass sie Wahrscheinlichkeiten für die verschiedenen Ausprägungen einer Nutzerintentionen gar nicht oder nur durch komplexe, für den Menschen schwer verständliche, Strukturen abbilden können (vgl. Anforderung A14). Auch mangelt es beiden an einer generellen Verständlichkeit des Modells (vgl. Anforderung A19). Diese Verständlichkeit ist im Rahmen insbesondere von Bedeutung, da Modellierer die Möglichkeit haben sollen, dass Modell manuell zu erstellen und zu bearbeiten. Für Modelle, die vollautomatisch und ohne menschliche Unterstützung gelernt werden würden, wäre auch der Einsatz von KNNs denkbar. Bayes'sche Netze hingegen erfüllen alle in Kapitel 3 ermittelten Anforderungen und werden daher als geeignetste Methode zur Modellierung von Nutzerintentionen für FIS eingestuft. Die weiteren Abschnitte und Kapitel der vorliegenden Arbeit werden daher zeigen, welche Prozesse und Implementierungen entwickelt wurden, um Bayes'sche Netze zur Modellierung von Nutzerintentionen für FIS zu verwenden.

[6] BNIF Website: http://www.cs.cmu.edu/~fgcozman/Research/InterchangeFormat/Old/xmlbif02.html. Zuletzt überprüft am 19.10.2016.

[7] XMLBIF Website: http://www.cs.cmu.edu/~fgcozman/Research/InterchangeFormat/. Zuletzt überprüft am 19.10.2016.

[8] Java Bayes Website: http://www.cs.cmu.edu/~javabayes/. Zuletzt überprüft am 19.10.2016.

[9] Weka Website: http://www.cs.waikato.ac.nz/ml/weka/index.html. Zuletzt überprüft am 19.10.2016.

	EB	KNN	BN
Anforderung A14: Wahrscheinlichkeit der Nutzerintention	–	○	+
Anforderung A15: Mehrere Nutzerintentionen zu einem Zeitpunkt	+	+	+
Anforderung A16: Nutzerintentionen sind situationsabhängig	+	+	+
Anforderung A17: Anpassung an den individuellen Nutzer	○	+	+
Anforderung A18: Gelernte Nutzerintentionen können vergessen werden	○	+	+
Anforderung A19: Hohe Verständlichkeit des Modells	○	–	+
Anforderung A20: Modell ist standardisiert	+	+	+
Anforderung A21: Vorhandensein einer Tool-Unterstützung	+	+	+

EB – Entscheidungsbäume; KNN – Künstliche Neuronale Netze; BN – Bayes'sche Netze

Tabelle 5.4: Vergleich von Entscheidungsbäumen, Künstlichen Neuronalen Netzen und Bayes'schen Netzen mit Bezug auf die in Kapitel 3 ermittelten Anforderungen.

5.4 Prozess zur Modellierung

Der im Rahmen dieser Arbeit vorgeschlagene Prozess zur Modellierung von Nutzerintentionen besteht aus drei Schritten, die im folgenden detailliert erklärt werden:

1. Modellierung der Eingangsgrößen (Situationen)
2. Modellierung der Ausgangsgrößen (Intentionen)
3. Modellierung der Beziehungen zwischen Eingangs- und Ausgangsgrößen

1. Modellierung der Eingangsgrößen (Situationen)

Der erste Schritt bei der Modellierung von Nutzerintention mit Hilfe von Bayes'schen Netzen besteht für den Modellierer darin, die für die zu modellierenden Nutzerintentionen relevanten Situationen als Knoten im Bayes'schen Netz anzulegen. Dabei muss der Modellierer keine Kenntnis darüber haben, welchen Ursprung die Situationen haben, oder aus welchen Kontextinformationen sie abgeleitet wurden. Im Hinblick auf das zu Beginn dieses Kapitels eingeführte Beispiel, würden daher Knoten für die Situationen `Drowsiness` und `AloneInCar` angelegt werden. Als mögliche Werte der Knoten legt der Modellierer dabei die möglichen Ausprägungen der Situationen fest. Für eben dieses Beispiel also `Drowsy` und `Awake` bzw. `Alone` und `Not Alone`.

Ähnlich wie bei der Modellierung von Kontextinformationen (vgl. Kapitel 4) werden zur Kennzeichnung der Knoten Annotationen benutzt. Die vom Modellierer erstellten Knoten, die Situationen repräsentieren, werden als *Input* annotiert, um zu signalisieren, dass diese Knoten zur Laufzeit mit realen Daten versorgt werden müssen, damit das Bayes'sche Netz seine Berechnungen durchführen kann.

Am Ende dieses ersten Schrittes hat der Modellierer ein Bayes'sches Netz erstellt, das lediglich Knoten und noch keinerlei Kanten beinhaltet. Diese werden dem Netz erst zu einem späteren Zeitpunkt hinzugefügt.

2. Modellierung der Ausgangsgrößen (Intentionen)

Nachdem die Situationen vom Modellierer als Knoten im Bayes'schen Netz angelegt wurden, können auf die gleiche Art und Weise auch Knoten für die zu erkennenden Nutzerintentionen angelegt werden. Für das fortlaufende Beispiels bedeutet dies, dass Knoten für die Nutzerintention `Genre` mit den drei möglichen Werten `Pop`, `Rock` und `Jazz` und ein Knoten für die Nutzerintention `Source` mit den Werten `Mobile`, `Radio` und `Internal` angelegt werden. Genau wie bei KNNs und anders als bei Entscheidungsbäumen, können vom Modellierer beliebig viele Nutzerintentionen in einem Bayes'schen Netz modelliert werden.

Die so erstellten Knoten werden vom Modellierer als *Output* annotiert, um zu signalisieren, dass diese Knoten zur Laufzeit ein Ergebnis im Sinne einer Nutzerintention und einer dazugehörigen Wahrscheinlichkeit bereithalten werden.

Am Ende dieses zweiten Schrittes besteht das Bayes'sches Netz aus allen notwendigen Knoten: den Situationen, repräsentiert durch als *Input* annotierte Knoten, und Nutzerintentionen, repräsentiert durch als *Output* annotierte Knoten.

3. Modellierung der Beziehungen

Der dritte Schritt, bei der Modellierung von Nutzerintentionen mit Hilfe Bayes'scher Netze, besteht für den Modellierer darin, Beziehungen zwischen Situationen (den Eingangsgrößen) und Nutzerintentionen (den Ausgangsgrößen) zu modellieren. Diese Beziehungen bestehen zum Einen aus einer Abhängigkeit, die angibt, welche Nutzerintentionen von welchen Situationen abhängen. Im Bayes'schen Netz werden diese Abhängigkeiten über gerichtete Kanten von der Situation zur Nutzerintention abgebildet. Zum Anderen bestehen die Beziehungen aus den Wahrscheinlichkeitstabellen, die angeben, wie wahrscheinlich eine bestimmte Nutzerintention unter Berücksichtigung der Ausprägungen der zugrundeliegenden Situationen ist. Für alle Kombinationen an Ausprägungen der für eine Nutzerintention relevanten Situationen, müssen vom Modellierer dabei die Wahrscheinlichkeiten für alle mögliche Ausprägungen dieser Nutzerintention angegeben werden. Die Summe dieser Wahrscheinlichkeiten muss dabei immer 1 ergeben, weswegen die Möglichkeit besteht nicht alle Wahrscheinlichkeiten anzugeben, da die verbleibenden berechnet werden können.

Am Ende dieser drei Schritte steht ein vollständiges Bayes'sches Netz, das Situationen, Nutzerintentionen und Beziehungen zwischen diesen beinhaltet. Wie die so erstellten Bayes'schen Netzen als Modell der zu erkennenden Nutzerintentionen zur Laufzeit benutzt werden können, um tatsächlich Nutzerintentionen aufgrund der aktuellen Situation zu erkennen, wird in Kapitel 6 detailliert beschrieben.

5.5 Zusammenfassung

In diesem Kapitel wurde untersucht, wie Nutzerintentionen für FIS modelliert werden können. Dazu wurden verschiedene Techniken und Methoden vorgestellt. In Abschnitt 5.1 wurde gezeigt, wie Entscheidungsbäume für eben diese Aufgabe genutzt werden können. Dabei wurde deutlich, dass Entscheidungsbäume zwar prinzipiell für die Modellierung von Nutzerintentionen geeignet sind, jedoch zum Beispiels Schwächen beim Umgang mit Wahrscheinlichkeiten von Intentionen aufweisen (vgl. Anforderung A14). Anschließend wurde in Abschnitt 5.2 untersucht, wie Künstliche Neuronale Netze für eine Modellierung von Nutzerintentionen genutzt werden können. Dabei zeigte sich, dass sie ebenfalls prinzipiell für diese Aufgabe geeignet sind. Es wurde jedoch auch gezeigt, dass die geforderte hohe Verständlichkeit des Modells (vgl. Anforderung A19), insbesondere für gelernte Künstliche Neuronale Netze, nicht gewährleistet werden kann. Daher wurde in Abschnitt 5.3 untersucht, wie geeignet Bayes'sche Netze im Hinblick auf die in Kapitel 3 ermittelten Anforderungen, zur Modellierung von Nutzerintentionen sind. Als Ergebnis dieser Untersuchung lässt sich festhalten, dass sich insbesondere die Anforderungen zum Umgang mit Wahrscheinlichkeiten und zur hohen Verständlichkeit des Modells, bei der Verwendung Bayes'scher Netze vollständig erfüllen lassen. In Abschnitt 5.4 wurde abschließend gezeigt, welcher Prozess zur Modellierung von Nutzerintentionen bei der Verwendung Bayes'scher Netze im Rahmen dieser Arbeit erarbeitet wurde.

6. Implementierung

„Most of the good programmers do programming not because they expect to get paid or get adulation by the public, but because it is fun to program.“

(Linus Torvalds, führender Entwickler von Linux und Git, [Gho98])

In diesem Kapitel wird gezeigt, wie die Modelle über Kontextinformationen und Nutzerintentionen genutzt werden können, um mithilfe vons Software-Komponenten Situationen und Nutzerintentionen auf Basis aktueller Sensorwerte zur Laufzeit erkennen zu können. Dazu wurden im Rahmen dieser Arbeit zum einen *generische* Software-Komponenten entwickelt, die unabhängig von Anwendungsfall, Kontextinformationen und den zu erkennenden Situationen und Nutzerintentionen sind. Die Implementierung dieser Komponenten ändert sich auch dann nicht, wenn sich das Modell der Kontextinformationen oder Nutzerintentionen ändert. Zum anderen wurden auch Software-Komponenten entwickelt, die eine Abhängigkeit zu einem oder beiden Modellen aufweist. Ihre Implementierung ändert sich daher bei einem sich verändernden Modell.

Dieses Kapitel ist wie folgt aufgebaut: zunächst werden die beiden Systeme zur Erkennung von Kontextinformationen und zur Erkennung von Nutzerintentionen getrennt voneinander in Abschnitt 6.1 und Abschnitt 6.2 vorgestellt. Dabei wird jeweils auf die generelle Funktionsweise sowie die dafür implementierten Software-Komponenten detailliert eingegangen. Im Anschluss wird in Abschnitt 6.3 vorgestellt, wie die nicht generischen Software-Komponenten beider Systeme auf Basis von EXLAP-Interfaces (vgl. Abschnitt 2.1.2.1), der Ontologie als Modell der Kontextinformationen und dem Bayes'schen Netz als Modell der Nutzerintentionen, vollständig generiert werden können. Dabei ist das vordergründige Ziel stets, den Prozess zur Erstellung des Gesamtsystems bei sich verändernden Modellen ohne manuelle Implementierungsaufwände zu gestalten. Abschnitt 6.4 fasst die in diesem Kapitel vorgestellten Arbeiten zusammen.

Die Inhalte dieses Kapitels wurden bereits in Teilen in [LSS15] veröffentlicht.

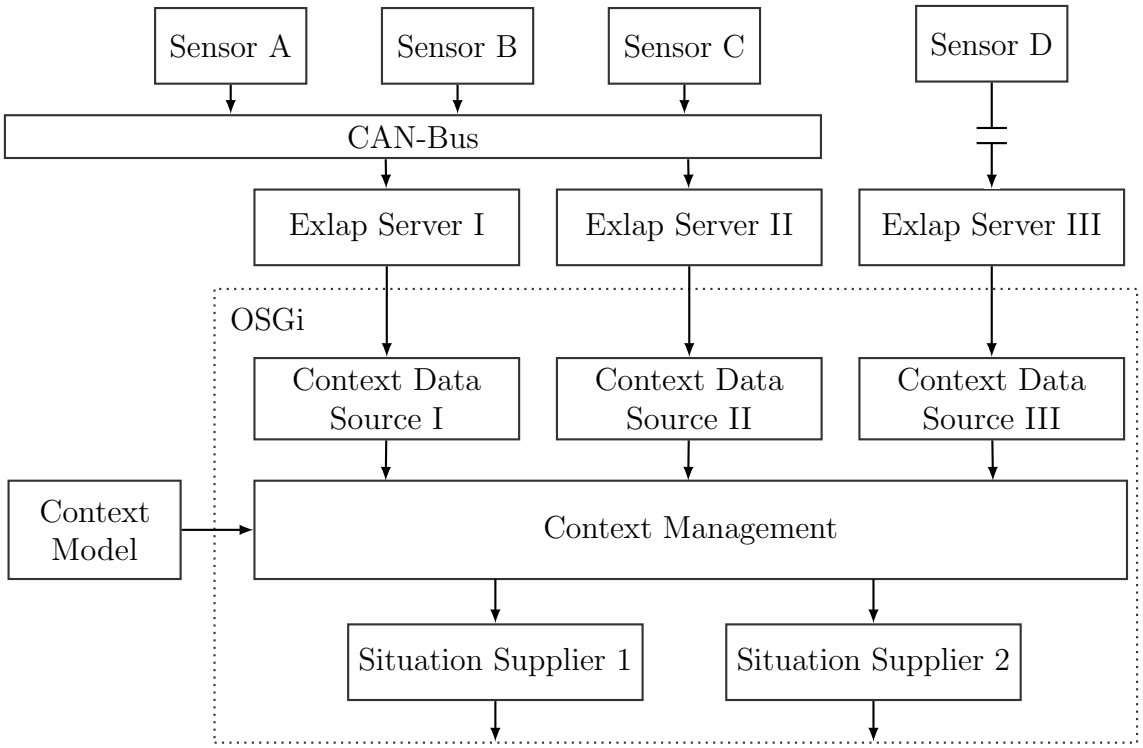

Abbildung 6.1: Schematische Darstellung des Datenflusses im System zur Auswertung von Kontextinformationen

6.1 Auswertung von Kontextinformationen

Basierend auf den im Modell erfassten Kontextinformationen, deren Beziehungen und den daraus resultierenden Situationen, soll zur Laufzeit die aktuelle Situation des Nutzers erfasst werden. Dazu wurde im Rahmen dieser Arbeit ein System entwickelt, das eine Ontologie zur Laufzeit mit den aktuellen Werten der Kontextinformationen befüllt und durch ein anschließendes Reasoning bewertet, in welcher Situation sich der Nutzer aktuell befindet.

Es wird zunächst ein Überblick über das Gesamtsystem und dessen generelle Funktionsweise gegeben, bevor detailliert auf die einzelnen Software-Komponenten eingegangen wird. Im Anschluss daran wird gezeigt, welche Auswirkungen ein verändertes Modell auf das Gesamtsystem hat und welche Schritte im Rahmen dieser Arbeit unternommen wurden, um diese zu minimieren.

6.1.1 Funktionsweise

Alle am System zur Auswertung von Kontextinformationen beteiligten Software-Komponenten und der dazugehörige Datenfluss sind in Abbildung 6.1 dargestellt. Sensoren stellen ihre ermittelten Werte über einen CAN-Bus des Fahrzeugs allen an diesem Bus angeschlossenen Geräten, Sub-Systemen und Software-Komponenten zur Verfügung. An den CAN-Bus sind auch EXLAP-Server (vgl. Abschnitt 2.1.2.1) angeschlossen. Sie greifen definierte Informationen vom CAN-Bus ab und stellen diese über das XML-basierte Netzwerkprotokoll EXLAP zur Verfügung. Dies bietet zum

einen den Vorteil eines einfachen Zugriffs auf die Daten, da nachgelagerte Systeme keine CAN-Frames interpretieren müssen, sondern direkt auf die Daten in Form ihres jeweiligen Datentyps zugreifen können. Zum anderen wird durch den Einsatz von EXLAP auch vom eigentlichen Transportmedium abstrahiert. So ist es beispielsweise auch möglich Daten nicht über den CAN-Bus, sondern über andere Bus-Systeme (wie zum Beispiel LIN, FlexRay, oder MOST) oder eine Online-Verbindung des Fahrzeugs anzubinden, ohne dabei nachgelagerte Software-Komponenten ändern zu müssen, da diese nach wie vor mit dem entsprechenden EXLAP-Server kommunizieren (vgl. Sensor D in Abbildung 6.1). Ausgangsbasis für alle weiteren Software-Komponenten sind daher EXLAP-Server, die Daten in einem lokalen Netzwerk über seine Socket-Verbindung zur Verfügung stellen. Weitere Software-Komponenten können sowohl auf der gleichen, als auch auf einer zusätzlichen Hardware implementiert sein, so lange diese über ein ein TCP/IP-Netzwerk mit der Hardware, auf der die EXLAP-Server implementiert sind, verbunden sind. Die Kommunikation von Daten vom Sensor bis zum EXLAP-Server ist nicht Teil dieser Arbeit, da die dabei verwendeten Komponenten und Beschreibungssprachen bereits zuvor zur prototypischen Entwicklung von Fahrzeugen eingesetzt wurden. Daher werden im weiteren Verlauf nur nachgelagerte Komponenten detailliert beschrieben.

Wie bereits in Abschnitt 2.1.2.2 beschrieben, wird in der Konzernforschung der Volkswagen AG bereits ein ein Open Services Gateway initiative (OSGi)-Framework eingesetzt, in dem sich ohne großen Aufwand neue Software-Komponenten etablieren lassen, um beispielsweise auf Fahrzeugdaten zuzugreifen. Die ihm Rahmen dieser Arbeit entwickelten Software-Komponenten sind daher Teil dieses Frameworks. Die entwickelten Konzepte, Datenstrukturen und konkreten Implementierungen lassen sich jedoch auch leicht in einem anderen Framework realisieren.

Für die Auswertung von Kontextinformationen sind daher, neben dem Modell der Kontextinformationen (der Ontologie), drei verschiedene Typen von Software-Komponenten verantwortlich:

a) *Context Data Source*, zum Empfang von Kontextinformationen, die von EXLAP-Servern bereitgestellt werden,

b) *Context Management*, welches die Ontologie mit empfangenen Kontextinformationen befüllt und durch ein anschließendes Reasoning ermittelt, in welcher Situation sich der Nutzer befindet und

c) *Situation Supplier*, um ermittelte Situationen an nachgelagerte Software-Komponenten zu kommunizieren.

Dem klassischen *EVA-Prinzip* [Dwo73] folgend, lassen sich diese drei Komponenten den Kategorien *Eingabe*, *Verarbeitung* und *Ausgabe* in gleicher Reihenfolge zuordnen. Da sie, wie bereits erwähnt, im Rahmen dieser Arbeit als Teil eines OSGi-Frameworks realisiert wurden, wird die gesamte Kommunikation zwischen diesen Komponenten vom Framework übernommen, weshalb auf die technischen Details dieser Kommunikation nicht weiter eingegangen wird.

6.1.2 Empfang von Kontextinformationen

Kontextinformationen werden von EXLAP-Servern bereitgestellt. Dabei ist es üblich, dass für verschiedene Arten von Kontextinformationen verschiedene EXLAP-Server existieren. Um ein möglichst hohes Maß an Modularität und Flexibilität zu gewährleisten, empfiehlt es sich für jeden EXLAP-Server eine Software-Komponente des Typs *Context Data Source* im System zu etablieren. Diese übernehmen die Aufgabe, die EXLAP-spezifischen Datenstrukturen und Datenformate, in ein von der Art der Kontextinformation unabhängiges Format zu übersetzen. Dazu verbinden sie sich mit dem EXLAP-Server, für den sie zuständig sind und registrieren sich bei diesem für alle Datenänderungen an Objekten, die im System zur Verarbeitung von Kontextinformationen relevant sind. Das Auslagern dieser Aufgabe in eine extra Software-Komponente und das Einführen einer generischen Schnittstelle zur Kommunikation von Kontextinformationen bringt zwei entscheidende Vorteile mit sich:

1. Bei Änderungen an der Schnittstelle zum *Context Management* müssen die EXLAP-Server nicht angepasst werden. Da auch weitere Software-Komponenten im Fahrzeug Daten der EXLAP-Server empfangen, würde eine enge Kopplung von EXLAP-Server und *Context Management* zu einer hohen Inflexibilität führen, da eine Änderung des *Context Managements* mit unter Änderungen in allen Komponenten zur Folge hätte, die mit einem EXLAP-Server kommunizieren.

2. Das Einführen neuer Kontextinformationen macht keine Änderungen am *Context Management* notwendig. Da die Schnittstelle, über die das *Context Management* generisch ist, können beliebig viele und semantisch verschiedene Kontextinformationen an das *Context Management* angebunden werden, ohne die Implementierung des *Context Managements* verändern zu müssen. Für einen neuen EXLAP-Server oder für neue Daten in einem bestehenden EXLAP-Server muss lediglich eine neue *Context Data Source* implementiert bzw. eine bestehende angepasst werden. Das *Context Management* kann dabei unverändert bleiben.

Das dabei eingesetzte, ebenfalls generische, Austauschformat beinhaltet den Namen der Quelle, aus der die Kontextinformationen entsprungen ist, den Namen der Klasse, mit der diese Art der Kontextinformation in der Ontologie modelliert wurde, sowie eine Menge an Datenobjekten, die die eigentlichen Werte der Kontextinformation beinhalten. Diese Datenobjekte wiederum enthalten einen Namen zur Identifikation des Wertes, die zu übertragenden Rohdaten, sowie den Datentyp des Wertes, um dem *Context Management* die Möglichkeit zu gewährleisten, die Rohdaten korrekt zu interpretieren. Im Rahmen dieser Arbeit wurde eine Unterstützung der folgenden Datentypen entwickelt: *Boolean*, *Date*, *Decimal*, *Long*, *Integer*, *String* und *Object*. Weitere Datentypen können bei Bedarf ergänzt werden. Die Datenstruktur zum Transport von Kontextinformationen sind in Anhang A dargestellt. Das generische Austauschformat ist als Java-Klasse in Quelltext A.1 gezeigt. Darin verwendete Typen und Java-Klassen finden sich in Quelltext A.2 und Quelltext A.3.

6.1.3 Verarbeitung von Kontextinformationen

Bevor die ersten Kontextinformationen dem *Context Management* zur Verfügung gestellt werden, führt dieses einige initialisierende Schritte durch, um später den korrekten Ablauf gewährleisten zu können. Dazu gehört unter anderem das Laden der Ontologie und das Durchsuchen nach Klassen, die mit Hilfe der *ExchangeType* Annotation als *Exchange_Out* gekennzeichnet wurden. Das sind Klassen, die Situationen repräsentieren und die später vom *Context Management* anhand von Kontextinformationen klassifiziert werden. Für die übergeordneten Klassen der so annotierten Klassen, werden vom *Context Management* Individuen angelegt. Notwendig ist dieser Schritt, um im weiteren Verlauf überprüfen zu können, ob Individuen während des Reasonings auch Subklassen zugeordnet wurden und somit signalisieren, dass eine bestimmte Situation vorliegt. Bezugnehmend auf das in Abschnitt 4.2 bereits eingeführte Beispiel würde das *Context Management* die Subklassen `No`, `Medium` und `High` finden, da diese vom Modellierer mit *Exchange_Out* annotiert wurden. Anschließend würde es ein Individuum der übergeordneten Klasse `Drowsiness` anlegen, das später dann eben diesen Subklassen zugeordnet werden kann.

Nach diesen initialen Schritten ist das *Context Management* bereit, Kontextinformationen von Software-Komponenten des Typs *Context Data Source* zu empfangen. Dafür bietet es die in Quelltext 6.1 als Java-Interface gezeigte Schnittstelle an. Neben einer Menge Kontextinformationen wird dabei auch der Wahrheitswert (Wahrscheinlichkeit für die Korrektheit dieses Wertes) für diese Kontextinformationen übermittelt.

Alle über diese Schnittstelle kommunizierten Kontextinformationen werden anschließend ebenfalls als Individuen innerhalb der Ontologie erstellt und bekommen den mitübertragenen Wahrheitswert zugewiesen. Bereits bestehende Individuen des gleichen Typs, werden mit den neuen Werten der Kontextinformation aktualisiert. Das *Context Management* kann diese Individuen anlegen bzw. aktualisieren, da innerhalb des generischen Austauschformats von der *Context Data Source* hinterlegt wurde, welcher Klasse der Ontologie diese Kontextinformation entspricht.

Die Ontologie beinhaltet zu diesem Zeitpunkt, neben den zur Entwicklungszeit erstellen Klassen und Beziehungen, sämtliche Kontextinformationen, die zur Verfügung gestellt wurden, sowie Individuen, die während der Initialisierung erstellt wurden und den zu erkennenden Situationstypen entsprechen. Anschließend wird das Reasoning der Ontologie durchgeführt. Dabei können verschiedene Reasoner zum Einsatz kommen, solange die im Modell verwendeten Techniken (wie zum Beispiel SWRL-Regeln) vom jeweiligen Reasoner unterstützt werden. Für der Entwicklung des

```
public interface CMContextConsumer {

    public void receiveContextData(List<CMIndividual> newIndividuals,
        double confidence);

}
```

Quelltext 6.1: Schnittstelle zum Empfang von Kontextinformationen als Java-Interface

```
public abstract interface DrowsinessListener {

    public abstract void onNo(double confidence);

    public abstract void onMedium(double confidence);

    public abstract void onHigh(double confidence);
}
```

Quelltext 6.2: Situationsspezifische Schnittstelle zur Kommunikation von Situationsinformationen als Java-Interface

hier vorgestellten Systems wurde der *Pellet-Reasoner*[SPG+07] eingesetzt. Während des Reasonings werden die modellierten Beziehungen und SWRL-Regeln ausgewertet. Als Ergebnis werden Individuen Klassen zugeordnet, denen sie vorher nicht zugeordnet waren. So könnte beispielsweise die Klasse `Drowsiness` auch der Subklasse `Medium` zugewiesen werden und signalisieren, dass der Nutzer einer leichten Müdigkeit unterliegt.

Nach dem Reasoning durchsucht das *Context Management* die von ihm angelegten Individuen und stellt fest, ob sie Klassen zugeordnet wurden, die mit *Exchange_Out* gekennzeichnet wurden. Anschließend berechnet das *Context Management* die Wahrheitswerte dieser Individuen. Dazu fragt es in einem ersten Schritt beim Reasoner an, mit welcher Kausalkette der Reasoner zu dem Ergebnis gekommen ist, dieses Individuum dieser Klasse zuzuordnen. Die vom Reasoner daraufhin gelieferte Begründung wird vom *Context Management* genutzt, um den Wahrheitswert dieser Zuordnung gemäß den vom Modellierer festgelegten Techniken (vgl. Abschnitt 4.2.3) zu ermitteln. Dieser Wahrheitswert wird als zusätzliches Attribut an der Klasse gespeichert.

In einem abschließenden Schritt, werden die mit *Exchange_Out* annotierten Klassen, zu denen nach dem Reasoning mindestens ein Individuum existierte, an die *Situation Supplier* weitergegeben. Dazu wurde ähnlich zum Empfang von Kontextinformationen, für die Ausgabe von Situationen eine generische Schnittstelle geschaffen, sodass die Implementierung des *Context Managements* für neu zu erkennende Situationen nicht abgeändert werden muss. Stattdessen kommuniziert das *Context Management* lediglich die erkannten Subklassen zu allen *Situation Suppliern*.

6.1.4 Ausgabe von Situationsinformationen

Die vom *Context Management* kommunizierten Subklassen, werden zunächst an alle im System verfügbaren *Situation Supplier* kommuniziert. Jeder von ihnen entscheidet selbst, für welche der Subklassen er zuständig ist. Allen gemein ist jedoch, dass sie für nachgelagerte Software-Komponenten eine situationsspezifische Schnittstelle anbieten, die von dieser nachgelagerten Komponenten implementiert werden muss und aus einfachen Methodenaufrufen zur Klassifizierung der jeweiligen Situation besteht. Im Hinblick auf das fortlaufende Beispiel würde ein *Situation Supplier* für die Situation `Drowsiness` so beispielsweise die in Quelltext 6.2 gezeigte Schnittstelle bedienen. Je nach aktueller Ausprägung würde er zum Beispiel die Methode

`onMedium(confidence)`, parametrisiert durch den ermittelten Wahrheitswert der Situation, an der nachgelagerten Software-Komponente aufrufen, wenn die Kontextinformationen darauf schließen lassen, dass der Nutzer des Systems leicht müde ist.

Wie auch für die Software-Komponenten vom Typ *Context Data Source*, muss der Entwickler des Systems zur Verarbeitung von Kontextinformationen sicherstellen, dass die benötigten *Situation Supplier* implementiert wurden und zur Laufzeit im System zur Verfügung stehen, damit sie vom *Context Management* gefunden und benachrichtigt werden können. Andernfalls können die erkannten Situationen nicht weiter verarbeitet werden.

6.2 Auswertung von Nutzerintentionen

Das System zur Erkennung von Nutzerintentionen, empfängt die erkannten Situationen und nutzt das in Kapitel 5 vorgestellte Modell, um auf Basis dieser Situationen, das FIS auf die aktuellen Bedürfnisse des Nutzers anzupassen.

Alle am System zur Auswertung von Nutzerintentionen beteiligten Software-Komponenten und der dazugehörige Datenfluss sind in Abbildung 6.2 dargestellt. Die bereits aus Abschnitt 6.1 bekannten Komponenten vom Typ *Situation Supplier* stellen Ausprägungen der erkannten Situationen zur Verfügung. Die weiteren Komponenten lassen sich, genau wie das System zur Auswertung von Kontextinformationen, in *Eingabe*, *Verarbeitung* und *Ausgabe* aufteilen. Die entsprechenden Software-Komponenten tragen die Namen *Situation Receiver*, *Intention Management* und *Intention Supplier* und werden im Folgenden detailliert vorgestellt. Auch diese drei Komponenten wurden als Teil des bereits erwähnten OSGi-Frameworks implementiert, weshalb das Beschreiben der Kommunikation erneut in Form von Java-Interfaces geschieht und auf weitere technische Details nicht näher eingegangen wird.

6.2.1 Empfang von Situationsinformationen

Wie bereits erwähnt, wird für jede zu erkennende Situation auf Basis der Ontologie eine Schnittstelle bereitgestellt, die von Software-Komponenten implementiert werden kann, die über Änderungen eben dieser Situation benachrichtigt werden möchten. Zur Erhöhung der Übersichtlichkeit und der besseren Modularität, wird an dieser Stelle empfohlen, die verschiedenen Schnittstellen der verschiedenen Situationen in eigenen Software-Komponenten zu implementieren. Als Ergebnis entsteht so eine Menge an *Situation Receiver*, von denen jeder die Änderungen einer bestimmten Situation vom entsprechenden *Situation Supplier* des Systems zur Asuwertung von Kontextinformationen mitgeteilt bekommt. Ein Beispiel für eine solche Schnittstelle wurde bereits in Quelltext 6.2 gezeigt.

Änderungen an der aktuellen Ausprägung einer Situation werden anschließend über das in Quelltext 6.3 gezeigte Austauschformat von Situationsinformationen *IMSituation* an das *Intention Management* weitergegeben. Dieses generische (für alle Situationen geltende) Austauschformat speichert den Namen und den Wert der

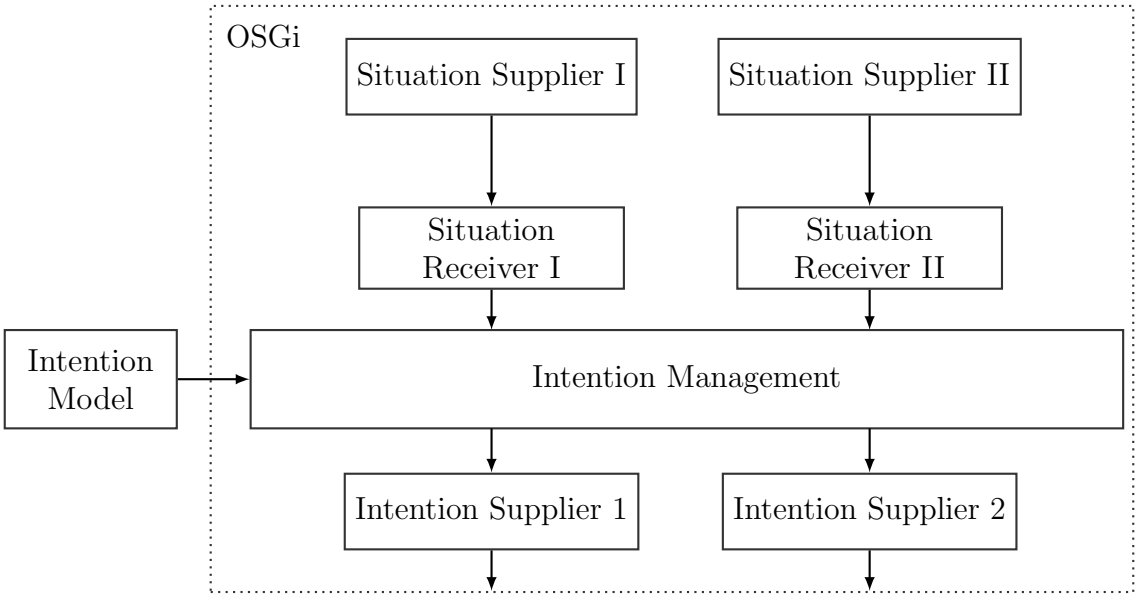

Abbildung 6.2: Schematische Darstellung des Datenflusses im Systems zur Auswertung von Nutzerintentionen

Situation als Zeichenkette und die Wahrscheinlichkeit als Fließkommazahl. Den Wert der Situation als Zeichenkette zu speichern, ist an dieser Stelle ausreichend, da Situationen zu diesem Punkt abstrahierte Werte, wie beispielsweise *Medium* oder *High* annehmen. Diese lassen sich ohne Einschränkungen als Zeichenketten speichern. Um die aktuell geltenden Situationen von allen *Situation Reciever* erhalten zu können, implementiert das *Intention Management* die in Quelltext 6.4 gezeigte Schnittstelle *IMSituationConsumer*. Es bietet die Möglichkeit, eine Liste von Situationen zu kommunizieren, welche von jedem *Situation Receiver* separat benutzt wird.

6.2.2 Verarbeitung von Situationsinformationen

Genau wie der *Context Manager* muss auch der *Intention Manager* einige initialisierende Schritte durchführen, bevor er Situationsinformationen von einem *Situation Recevier* empfangen kann. Zunächst lädt er das Modell der Nutzerintentionen (das Bayes'sche Netz) und durchsucht dessen annotierte Knoten, um festzustellen, welche mit *Input* und welche mit *Output* annotiert wurden. Die mit *Input* annotierten Knoten, entsprechen Situationen und sind daher für den Empfang von Situationsiformationen von erhöhter Bedeutung. Mit *Output* annotierte Knoten hingegen, sind bei der Ausgabe von Intentionen zu beachten.

Empfängt der *Intention Manager* zur Laufzeit Situationsinformationen von einem *Situation Receiver*, so stellt er den Wert des Knoten im Bayes'schen Netz entsprechend des übermittelten Wertes ein. Wurden alle mit *Input* annotierten Knoten mit Werten befüllt, kann das Bayes'sche Netz genutzt werden, um die Wahrscheinlichkeitsverteilungen für die mit *Output* annotierten Knoten zu berechnen. Als Ergebnis ergibt sich für jede Ausprägung aller Nutzerintentionen eine Wahrscheinlichkeit die angibt, ob der Nutzer aktuell die jeweilige Intention hat. Basierend auf dem Beispiel aus Kapitel 5, würden sich so Wahrscheinlichkeitsverteilungen für die Intentionen

```
public class IMSituation {

    private String situationName;
    private String situationValue;
    private double confidence;

    public IMSituation(String situationName, String situationValue,
        double confidence) {
        this.situationName = situationName;
        this.situationValue = situationValue;
        this.confidence = confidence;
    }

    public String getSituationName() {
        return this.situationName;
    }

    public String getSituationValue() {
        return this.situationValue;
    }

    public double getConfidence() {
        return this.confidence;
    }
}
```

Quelltext 6.3: Generisches Format zum Austausch von Situationsinformationen als Java-Klasse

`Genre` und `Source` ergeben, aus denen erkenntlich ist, welches Musikgenre der Nutzer hören und welche Wiedergabequelle er benutzen möchte.

Diese Wahrscheinlichkeitsverteilungen werden in einem abschließenden Schritt an die *Intention Supplier* weitergeben. Dazu wird die in Quelltext 6.5 gezeigte Schnittstelle *IMSupplyDataInterface*, als auch das in Quelltext 6.6 gezeigte, generische Austauschformat für Intentionen *IMIntention* verwendet. Letzteres ermöglicht es, zu einer Intention (identifiziert über einen Namen) und zu jeder ihrer Ausprägungen eine Wahrscheinlichkeit zu speichern. Ein *Intention Supplier*, der diese Informationen vom *Intention Management* über die Schnittstelle *IMSupplyDataInterface* mitgeteilt bekommt, kann somit das Systemverhalten aufgrund der wahrscheinlichsten Ausprägung einer Intention ändern. Mit Bezug auf das in Kapitel 5 eingeführte Beispiel, könnte der für die Intention `Genre` zuständige *Intention Supplier* beispielsweise Musik mit einem bestimmten Genre abspielen.

6.2.3 Ausgabe von Intentionen

Analog zu Software-Komponenten des Typs *Situation Supplier*, verwenden *Intention Supplier* eine intentionsspezifische Schnittstelle, um die in der aktuellen Situation wahrscheinlichste Ausprägung einer Intention inklusive der dazugehörigen Wahrscheinlichkeit an nachgelagerte Komponenten zu kommunizieren. Zusätzlich informiert er

```
import java.util.List;

public interface IMSituationConsumer {

    public void onSituationChanged(List<IMSituation> situations);

}
```

Quelltext 6.4: Schnittstelle zum Austausch der aktuellen Situationsinformationen zwischen Situation Receiver und Intention Management

```
public abstract interface IMSupplyDataInterface {

    public abstract void onIntentionChanged(List<IMIntention>
        intentions);

}
```

Quelltext 6.5: Schnittstelle zum Austausch der aktuellen Nutzerintentionen als Java-Interface

auch noch über die Wahrscheinlichkeitsverteilung, indem er alle Ausprägungen mit ihrer entsprechenden Wahrscheinlichkeit kommuniziert. Eine exemplarische Schnittstelle für die Intention `Genre` ist in Quelltext 6.7 dargestellt.

Auch wenn es auf den ersten Eindruck redundant wirken mag, die Wahrscheinlichkeitsverteilung und dedizierte Methoden für die einzelnen Ausprägungen der Intention als Teil der Schnittstelle zur Verfügung zu stellen, so liegt der Vorteil darin, dass Entwickler nachgelagerter Komponenten sowohl eine einfache Möglichkeit (dedizierte Methoden), als auch detaillierte Möglichkeit (Wahrscheinlichkeitsverteilung) haben, die entsprechende Intention zu bewerten.

Das tatsächliche Anpassen des FIS erfolgt in nachgelagerten Software-Komponenten. Welche Funktionalitäten dabei in welcher Form verändert werden, hängt stark von den umgesetzten Anwendungsfällen ab und ist daher nicht Teil der im Rahmen dieser Arbeit entwickelten Systeme.

6.3 Prozess zur Erstellung des Gesamtsystems

Die in Abschnitt 6.1 und Abschnitt 6.2 vorgestellten Systeme zur Auswertung von Kontextinformationen und Nutzerintentionen beinhalten auf der einen Seite generische Software-Komponenten und Schnittstellen, wie zum Beispiel das *Context Management* oder das Austauschformat für Kontextinformationen, an denen keinerlei Änderungen bei geänderten Modellen vorgenommen werden müssen. Auf der anderen Seite arbeiten in beiden Systemen jedoch auch Software-Komponenten, die abhängig von modellierten Kontextinformationen bzw. Nutzerintentionen sind.

```
import java.util.HashMap;
import java.util.Map;

public class IMIntention {

    private String              intentionName;
    private Map<String, Double> probabilityDistribution;

    public IMIntention(String intentionName) {
        this.intentionName = intentionName;
        this.probabilityDistribution = new HashMap<String, Double>();
    }

    public void addIntentionValue(String value, double probability) {
        this.probabilityDistribution.put(value, probability);
    }

    public String getName() {
        return this.intentionName;
    }

    public Map<String, Double> getProbabilityDistribution() {
        return this.probabilityDistribution;
    }
}
```

Quelltext 6.6: Generisches Format zum Austausch von Nutzerintentionen als Java-Klasse

In diesem Abschnitt wird präsentiert, wie diese Software-Komponenten vollautomatisch unter Zuhilfenahme der beiden Modelle (Ontologie und Bayes'sches Netz) generiert werden können, so dass Änderungen im Modell keinen manuellen Implementierungsaufwand nach sich ziehen. Zusätzlich dazu wird auch gezeigt, wie Teile der Ontologie und des Bayes'schen Netzes generiert werden können, um den Modellierer bei seiner Tätigkeit zu unterstützen und die Fehleranfälligkeit zu minimieren. Die nun folgenden Abschnitten erklären detailliert und in chronologischer Reihenfolge, wie der im Rahmen dieser Arbeit entwickelte Prozess zur Erstellung des Gesamtsystems durch Code-Generierung unterstützt werden kann.

Die in Abbildung 6.3 dargestellten Beziehungen zeigen, welche Software-Komponenten aus welchen Modellen generiert werden können und für welche Modelle sich Vorlagen erstellen lassen, um den Modellierer maximal zu unterstützen.

6.3.1 Generieren der EXLAP-Server

Das Generieren der EXLAP-Server basiert auf den EXLAP-Interfaces (vgl. Abschnitt 2.1.2.1), die beschreiben welche Daten ein EXLAP-Server zur Verfügung stellt. Da die Generierung bereits vor dem Beginn der in dieser Arbeit präsentierten Entwicklung verfügbar war, jedoch wichtig zum Verständnis des Gesamtprozesses ist, sei es an dieser Stelle erwähnt, aber nicht detailliert beschrieben.

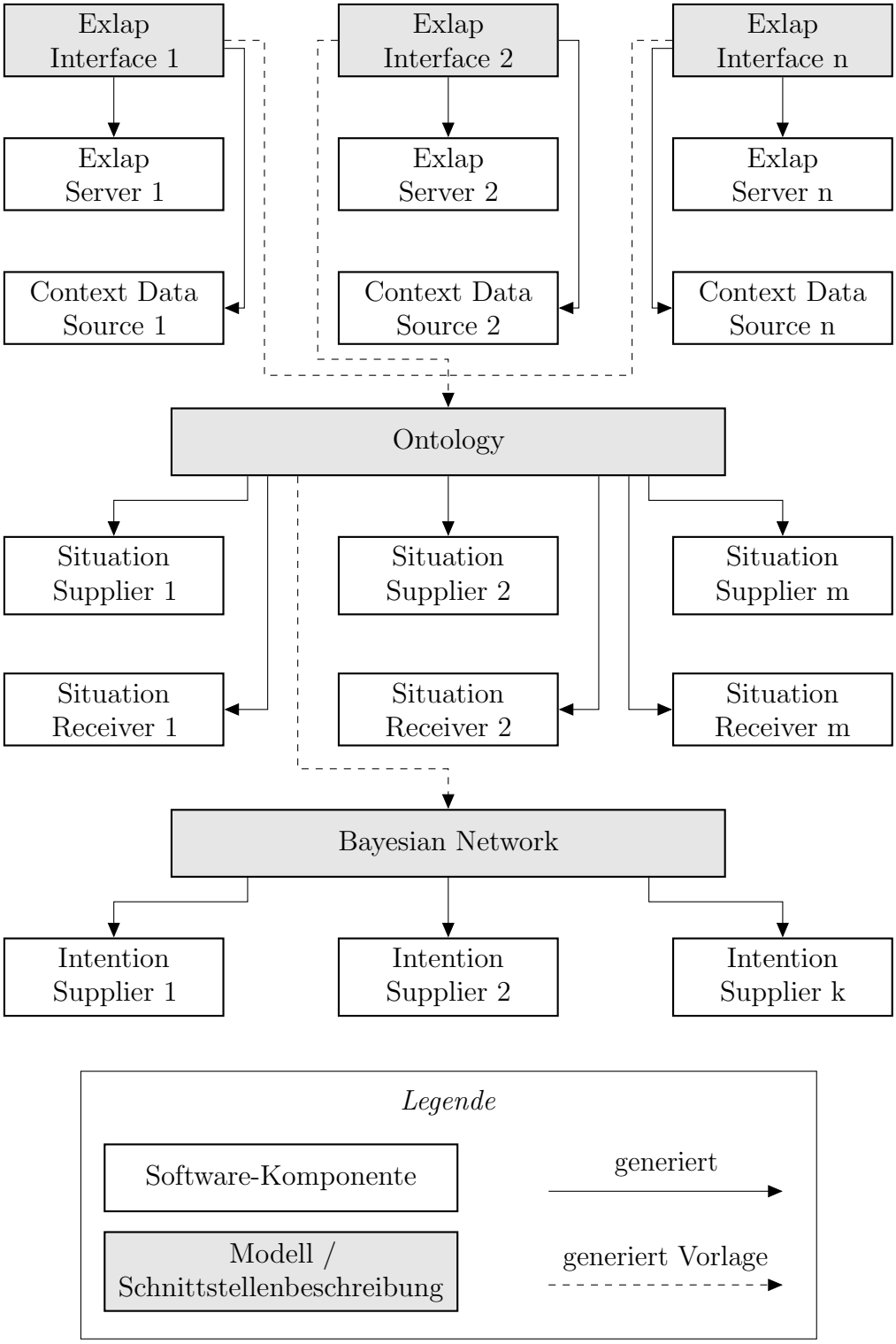

Abbildung 6.3: Übersicht der Code-Generierung

```
import java.util.Map;

public abstract interface GenreListener {

    public abstract void onRock(double probability);

    public abstract void onPop(double probability);

    public abstract void onJazz(double probability);

    public abstract void onChange(
                    Map<String, Double> probabilityDistribution);
}
```

Quelltext 6.7: Situationsspezifische Schnittstelle zur Kommunikation von Situationsinformationen als Java-Interface

Entscheidend ist, dass die Software-Komponente, welche die Kontextinformationen anbieten, auf Basis einer Schnittstellen-Beschreibung und ohne menschliches Zutun generiert werden können.

6.3.2 Generieren der Ontologie

Als zweiten Schritt können die gleichen EXLAP-Interfaces verwendet werden, um eine *Ontologie-Vorlage* zu erstellen. Dazu wird eine Ontologie generiert, die bereits Klassen beinhaltet, die den Datenobjekten der EXLAP-Interfaces entsprechen und die mit folgenden Informationen annotiert sind:

- *ExchangeType* Annotation wird mit *Exchange_In* gefüllt
- *SourcenName* Annotation wird mit dem Namen des EXLAP-Interfaces gefüllt
- *SourcePropertyName* Annotation wird mit dem Namen des Datenobjekts gefüllt
- *SourceType* Annotation wird mit *Source_Push* gefüllt

Die dabei entstehende Ontologie wird vom Modellierer als Vorlage für die weitere Modellierung verwendet. Dadurch wird ausgeschlossen, dass beim Anlegen der Eingangsgrößen Fehler, zum Beispiel in Form von Tippfehlern, gemacht werden, die erst in der Integrationsphase des Systems bemerkt werden würden. Zusätzlich stellt dieser Schritt für den Modellierer je nach Menge der Eingangsgrößen eine nicht zu vernachlässigende Aufwands- und Zeitersparnis dar.

6.3.3 Generieren der Context Data Sources

Auch die *Context Data Sources* können aus den EXLAP-Interfaces generiert werden. Ihre Aufgabe – die Umwandlung der von EXLAP-Servern empfangenen Daten in das generische Austauschformat von Kontextinformationen – folgt unter Berücksichtigung

der umzuwandelnden Kontextinformationen immer dem gleichen Schema. Dazu zählt: das Verbinden mit dem EXLAP-Server, das Registrieren an diesem, um über Änderungen an Datenobjekten benachrichtigt zu werden, das Umwandeln der Daten in das generische Format und das Kommunizieren dieser Daten zum *Context Management*. Unterschiedliche Kontextinformationen erfordern daher lediglich die Verbindung zu einem anderen EXLAP-Server und eine dem Datentyp der Datenobjekte angepasste Umwandlung. Um sich zur Laufzeit mit dem richtigen EXLAP-Server zu verbinden, wird lediglich dessen Name benötigt. Dieser und die Datentypen der Datenobjekte sind Teil des EXLAP-Interfaces, weshalb zu jedem leicht eine *Context Data Source* generiert werden kann.

Der Entwickler muss sicherstellen, dass alle benötigen Kontextinformationen (d.h. alle im Modell verwendeten Kontextinformationen) zur Laufzeit auch tatsächlich vorhanden sind. Nur dann kann gewährleistet werden, dass die ebenfalls im Modell hinterlegten Situationen auch vom System zur Verarbeitung von Kontextinformationen erkannt werden können.

6.3.4 Generieren der Situation Supplier

Während der Modellierung wurde mit Hilfe der Ontologie definiert, welche Situationen das System später erkennen können soll. Die dazu angelegten Klassen wurden mit *Exchange_Out* annotiert. Diese Information kann genutzt werden, um sowohl die *Situation Supplier*, als auch die situationsspezifische Schnittstelle, die sie bedienen (vgl. Abschnitt 6.1.4), zu generieren. Die dafür notwendigen Informationen sind der Klassenname und die Subklassen zur Beschreibung der Ausprägungen der Situation. Die Übersetzung aus dem generischen Format zur Übertragung von Situationsinformationen in die situationsspezifische Schnittstelle erfolgt immer nach dem gleichen Schema, weshalb die *Situation Supplier* ohne weitere Aufwände automatisch generiert werden können.

6.3.5 Generieren der Situation Receiver

Auch die *Situation Receiver* können auf Basis der Ontologie generiert werden. Sie übersetzen das situationsspezifische Format der *Situation Supplier* in das generische Format des *Intention Managements*. Auch diese Übersetzung folgt einem immer gleichen Schema, so dass passend zu den in der Ontologie modellierte Situationen die passenden *Situation Receiver* generiert werden können. Genau wie bei einem *Situation Supplier* werden dazu Klassen der Ontologie als Grundlage genommen, die während der Modellierung mit *Exchange_Out* annotiert wurden. Für jede dieser Klassen wird ein *Situation Receiver* generiert. Der so generierte Code implementiert die situationsspezifische Schnittstelle, die vom *Situation Supplier* bedient wird. Da diese bereits im vorherigen Schritt generiert wurde, kann sie auch ohne weiteres erneut generiert werden.

6.3.6 Generieren des Bayes'schen Netzes

Analog zur Generierung einer Ontologie-Vorlage kann eine Vorlage für das Bayes'sche Netz generiert werden. Dabei wird erneut die Ontologie als Grundlage herangezogen. Da in ihr alle zu erkennenden Situationen modelliert sind, wird für jede dieser zu

modellierten Situationen ein Knoten im Bayes'schen Netz generiert. Die Werte, die dieser Knoten annehmen kann, entsprechen erneut den Subklassen der mit *Exchange_Out* annotierten Klassen. Wie schon bei der Ontologie-Vorlage, nimmt dieser Schritt dem Modellierer nicht nur Arbeit ab, sondern wirkt sich auch positiv auf die Fehleranfälligkeit aus, da der Modellierer beim Anlegen der Knoten, die Situationen im Bayes'schen Netz repräsentieren, nun keine Fehler mehr machen kann.

6.3.7 Generieren der Intention Supplier

Nachdem der Modellierer auch das Bayes'sche Netz mit Inhalten befüllt hat, werden in einem letzten Schritt die notwendigen *Intention Supplier* generiert. Dazu wird das Bayes'sche Netz nach Knoten durchsucht, welche mit *Output* annotiert wurden. Diese Knoten repräsentieren Nutzerintentionen und müssen – analog zu den *Situationen* – von einem generischen Austauschformat, in eine intentionsspezifische Schnittstelle übersetzt werden.

6.4 Zusammenfassung

In diesem Kapitel wurde gezeigt, wie Systeme implementiert werden können, welche die Modelle von Kontextinformationen und Nutzerintentionen verwenden, um zur Laufzeit zu erfassen, in welcher Situation sich der Nutzer befindet und welche Intention er in dieser Situation hat. Beide Systeme wurde im Rahmen dieser Arbeit als Teil eines bereits bestehenden OSGi-Frameworks umgesetzt, was für die generelle Funktionalität und die Aufteilung der Komponenten jedoch nicht zwingend erforderlich ist. Die beiden zentralen Komponenten *Context Management* und *Intention Management* verwenden erst zur Laufzeit das jeweilige Modell und müssen für unterschiedliche Modelle nicht angepasst werden. Alle anderen beteiligten Software-Komponenten sind spezifisch mit Bezug auf den jeweiligen Anwendungsfall. Um den Implementierungsaufwand so gering wie möglich zu halten, wurde in diesem Kapitel auch gezeigt, wie der Quellcode dieser Komponenten basierend auf den Modellen generiert werden kann. Als Ergebnis muss bei einem veränderten Modell keine einzige Zeile Quellcode manuell geschrieben werden, da alle Komponenten entweder generisch (also unabhängig vom Modell) sind, oder generiert werden können.

7. Evaluation

„Objektivität: Alles hat zwei Seiten.
Aber erst wenn man erkennt, dass es drei sind,
erfasst man die Sache.“

(Heimito von Doderer, österreichischer Schriftsteller, [vW96])

In diesem Kapitel wird anhand von drei Forschungsfragen untersucht, wie geeignet die im Rahmen der vorliegenden Arbeit vorgestellten Techniken zur Modellierung von Kontextinformationen und Nutzerintentionen und die dazu implementierten Systeme zur Auswertung der Modelle sind. Die drei Forschungsfragen lauten:

1. Erfüllen Ontologien und Bayes'sche Netze die von Domänenexperten aufgestellten Anforderungen?
2. Welche Auswirkung hat eine steigenden Anzahl an modellierten Kontextinformationen und Situationen innerhalb einer Ontologie auf die Geschwindigkeit ihrer Auswertung?
3. Welche Auswirkung hat die Verwendung von SWRL-Regeln zur Modellierung von Kontextinformationen innerhalb einer Ontologie auf die Geschwindigkeit ihrer Auswertung?

Zur Beantwortung der ersten Forschungsfrage, wird in Abschnitt 7.1 zunächst die Fallstudie Intention-aware In-Car Infotainment System (INIS) vorgestellt und beschrieben. Auf Basis der Beobachtungen und der gesammelten Erfahrungen während der Entwicklung von INIS, werden anschließend in Abschnitt 7.2 die in Kapitel 3 ermittelten Anforderungen auf ihre Erfüllung hin untersucht. Zur Beantwortung der zweiten und dritten Forschungsfrage werden in Abschnitt 7.3 Erweiterungen, Messungen und Ergebnisse vorgestellt, die zeigen, welche Auswirkungen eine steigende Anzahl von Kontextinformationen und Situationen in einer Ontologie und

die Verwendung von SWRL-Regeln haben. Die Validität der Ergebnisse wird in Abschnitt 7.4 diskutiert, bevor Abschnitt 7.5 das Kapitel und die Antworten auf die Forschungsfragen abschließend zusammenfasst.

Die in diesem Kapitel vorgestellten Untersuchungen und Bewertungen wurden bereits in [LSSS15], [LSS15] und [LSSS16] veröffentlicht.

7.1 Fallstudie INIS

Mit Hilfe von INIS ist ein umfangreiches, kontext- und intentionssensitives FIS entstanden, dass aus einfachen Sensorinformationen, wie zum Beispiel der Tageszeit, dem Wochentag oder dem Tankfüllstand des Fahrzeugs, Intentionen seines Nutzers ableiten kann. INIS wird in diesem Kapitel jedoch nicht inhaltlich bewertet. Stattdessen werden die im Rahmen dieser Arbeit vorgeschlagenen Techniken zur Modellierung von INIS evaluiert. Es wird daher weder auf die Sinnhaftigkeit der modellierten Situationen, noch auf den Nutzen der modellierten Intentionen näher eingegangen.

Die Domänenexperten näherten sich bei der Erstellung von INIS der Aufgabenstellung in einem *top-down*-Ansatz, da sie zunächst das Endergebnis – die zu erkennenden Intentionen – modellierten und sich von dort aus über die dafür erforderlichen Situationen bis hin zu den wiederum erforderlichen Kontextinformationen vorarbeiteten. In den kommenden Abschnitten werden daher die Zwischenergebnisse dieser Arbeiten in gleicher Reihenfolge vorgestellt.

7.1.1 Modellierung von Nutzerintentionen

Um INIS ausreichend umfangreich zu gestalten, formulierten die Domänenexperten drei zu erkennende Nutzerintentionen. Ein möglichst vielfältiger Einsatz von INIS wurde von ihnen durch die Verteilung dieser drei Intentionen in die drei Bereiche Medien, Telefonie und Navigation des FIS gewährleistet. Die modellierten Nutzerintentionen umfassen:

- die bereits aus dem fortlaufenden Beispielen bekannte Nutzerintention *Genre*, die angibt welches Musikgenre der Nutzer aktuell hören möchte,
- die Nutzerintention *Calling Contact*, die angibt welchen Kontakt der Nutzer anrufen möchte und
- die Nutzerintention *POI Search*, die angibt, nach welchem Point of Interest (POI) (engl. für interessanten Ort), der Nutzer im Navigationssystem suchen möchte.

Die Modellierung dieser Intention geschah dabei, wie in Kapitel 5 vorgestellt, mit Hilfe Bayes'scher Netze.

Nutzerintention Genre

Die Nutzerintention *Genre* wurde, im Vergleich zum fortlaufenden Beispiel, von den Domänenexperten um weitere Ausprägungen ergänzt. Für INIS umfasst sie die Ausprägungen: *Blues*, *Classical*, *Country*, *Disco*, *HipHop*, *Jazz*, *Metal*, *Pop*, *Rock* und *Raggae*. Ihre eigentliche Aussage blieb hingegen unverändert. Sie gibt nach wie vor an, welches Musikgenre der Nutzer in der aktuellen Situation hören möchte.

Nutzerintention Calling Contact

Neu hinzugekommen ist die Nutzerintention *Calling Contact*. Sie soll angeben, welchen seiner Kontakte der Nutzer in der aktuellen Situation anrufen möchte. Typischerweise sind die möglichen Ausprägungen dieser Intention Kontakte im Adressbuch der Nutzer und damit stark nutzerspezifisch. Da eine Anpassung der Ausprägungen von Intentionen an individuelle Nutzer mit den im Rahmen dieser Arbeit vorgeschlagenen Techniken nicht vorgesehen war, wurden die Ausprägungen dieser Intention auf das Anrufen von bestimmten Favoriten gesetzt. Das FIS bietet die Möglichkeit, bis zu drei Kontakte aus dem Adressbuch als Favorit zu speichern. Diese können mit nur einer Nutzerinteraktion angerufen werden. Die Ausprägungen der Nutzerintention *Calling Contact*, wurde daher von den Domänenexperten auf *Favorite 1*, *Favorite 2*, *Favorite 3* und *None* festgelegt. Sie geben an, ob der Nutzer den Favoriten mit der ID 1, 2 oder 3 anrufen möchte, oder ob der in der aktuellen Situationen überhaupt keinen Favoriten anrufen möchte (Ausprägung *None*).

Nutzerintention POI Search

Teil des FIS ist auch ein Navigationssystem, dass es erlaubt nach POIs einer bestimmten Kategorie zu suchen. So kann der Nutzer zum Beispiel nach Tankstellen suchen. Die Nutzerintention *POI Search* gibt an, nach welcher Kategorie von POIs der Nutzer in seinem Navigationssystem in der aktuellen Situation suchen möchte. Dabei wurden von den Domänenexperten während der Modellierung des Systems die folgenden drei Ausprägungen definiert: *Gas*, die angibt, dass der Nutzer nach einer Tankstellen suchen möchte, *Parking*, die angibt, dass der Nutzer nach einem Parkplatz, einem Parkhaus, oder allgemein einer Parkmöglichkeit suchen möchte, und *None*, die angibt, dass der Nutzer in der aktuellen Situation nach keiner bestimmten POI-Kategorie suchen möchte.

Am Ende dieses ersten Schrittes liegt ein Bayes'sches Netz vor, dessen Knoten die beschriebenen Nutzerintentionen repräsentieren. Die möglichen Werte dieser Knoten sind dabei durch die möglichen Ausprägungen der Intentionen vorherbestimmt. Der nächste Schritt der Modellierung sieht vor, dass der Modellierer die zur Erkennung dieser Nutzerintentionen notwendigen Situationen dem Bayes'schen Netz als Knoten hinzufügt.

7.1.2 Modellierung erforderlicher Situationen

Nachdem die Domänenexperten die drei zu erkennenden Nutzerintentionen mit Hilfe Bayes'scher Netze modelliert hatten, konnten die dafür erforderlichen Situationen dem gleichen Netz als Knoten hinzugefügt werden. Dabei wurde definiert, dass die zuvor genannten Intentionen von acht Situationen abhängig sind, die im Folgenden näher erläutert werden. Zu diesem Zeitpunkt wurden von den Domänenexperten zwar die jeweiligen Ausprägungen der Situationen festgelegt, jedoch nicht unter welchen Umständen diese jeweiligen Ausprägungen gelten. Eine solche Festlegung geschieht im nächsten Schritt, wenn die modellierten Situationen in Abhängigkeit zu modellierten Kontextinformationen gesetzt werden.

Weather

Die Situation *Weather* wurde von den Domänenexperten mit den beiden Ausprägungen *Good*, um gutes, und *Bad*, um schlechtes Wetter repräsentieren zu können, versehen. Auch diese Situation wurde von den Domänenexperten mit einer Abhängigkeit zur Intention *Genre* modelliert.

Day Type

Die Situation *Day Type* dient zur einfachen Klassifizierung des Tagestyps. Dabei kann mit Hilfe der beiden Ausprägungen *Weekday* und *Weekend* zwischen einem Tag in der Woche und einem Tag am Wochenende unterschieden werden. Ausgehend von dieser Situation wurde eine Abhängigkeit zur Intention *Genre* von den Domänenexperten modelliert.

Day Time

Die Situation *Day Time* ermöglicht es mit ihren vier Ausprägungen die aktuelle Tageszeit zu klassifizieren. Zur Verfügung stehen dabei: *Morning* für eine Uhrzeit früh am Tage, *Noon* für eine Uhrzeit um die Mittagszeit, *Afternoon* für eine Uhrzeit am Nachmittag und *Night* für eine Uhrzeit in der Nacht.

Passenger

Die Situation *Passenger* wurde von den Domänenexperten im Bayes'schen Netz modelliert, um eine Möglichkeit zu haben, Nutzerintentionen in Abhängigkeit zu den im Fahrzeug befindlichen Passagieren zu modellieren. So bietet sie mit ihren Ausprägungen *None*, *Children*, *Friends* und *Other* die Möglichkeit, zu überprüfen, ob sich keine Passagiere im Fahrzeug befinden oder ob die Passagiere der Klasse Kinder, Freunde oder anderen Passagieren angehören. Für die Fallstudie INIS modellierten die Domänenexperten eine Abhängigkeit zu den Intentionen *Genre* und *Calling Contact*.

Driving To

Zur Klassifizierung des Ziels einer Fahrt mit einem mit INIS ausgestatteten Fahrzeuges modellierten die Domänenexperten die Situation *Driving To*. Dabei legten sie die Ausprägungen *Home*, *Work* und *Other* an, um auszudrücken, dass das Fahrzeug auf dem Weg nach Hause, zur Arbeit oder zu einem anderen Ort ist. Diese Ausprägungen wirken sich durch die modellierten Abhängigkeiten auf die Nutzerintentionen *Calling Contact* und *POI Search* aus.

Fuel Level

Die modellierte Situation *Fuel Level*, ermöglicht es durch ihre Ausprägungen *Low*, *Medium* und *High*, Nutzerintentionen bei einem niedrigen, mittleren oder hohen Füllstand des Tanks oder der Batterie des Fahrzeugs, verschieden auszuprägen. Von den Domänenexperten wurde modelliert, dass die Nutzerintention *POI Search* von dieser Situation abhängig ist.

Near To

Die letzte modellierte Situation *Near To*, beschreibt die örtliche Nähe des Fahrzeugs zu markanten Punkten. Um die Nähe zu den Punkten der Heimat- und Arbeitsadresse des Nutzer als auch zu anderen Punkten ausdrücken zu können, modellierten die Domänenexperten die Ausprägungen *Home*, *Work* und *Other*. Außerdem wurden Abhängigkeiten zu den Intentionen *POI Search* und *Calling Contact* dem Modell hinzugefügt.

Die beschriebenen Situationen wurden im Bayes'schen Netz ebenfalls als Knoten modelliert. Die Abhängigkeiten zwischen Situationen und Nutzerintentionen wurden wie in Kapitel 5 beschrieben als Kanten zwischen den entsprechenden Knoten modelliert. Zu jedem Knoten, der eine Intention repräsentiert, wurde im Anschluss eine Wahrscheinlichkeitsverteilung angegeben. Für den Knoten `POI Search` ist eine solche Verteilung exemplarisch in Tabelle 7.1 dargestellt.

Das Ergebnis dieses zweiten Schritts, in dem von den Domänenexperte Situationen und die Abhängigkeiten zwischen diesen und den Nutzerintentionen modelliert wurden, ist gemäß Kapitel 5 ein Bayes'sches Netz, das Knoten für Situationen, Nutzerintentionen und Abhängigkeiten in Form von bedingten Wahrscheinlichkeiten enthält. Die Struktur des Bayes'schen Netzes für INIS ist in Abbildung 7.1 dargestellt. Für die Knoten `Genre`, `Calling Contact` und `POI Search`, wurden anschließend

7.1.3 Modellierung erforderlicher Kontextinformationen

Auf Basis der zuvor modellierten Situationen kann im Anschluss modelliert werden, aus welchen Kontextinformationen sich diese gegebenen Situationen ableiten lassen. Dazu wurden, wie in Kapitel 4 beschrieben, Ontologien und die ebenfalls vorgestellten Erweiterungen verwendet. Am Beispiel der Situation `Weather` wird verdeutlicht, welchem Muster diese Modellierung folgt. Der für diese Situation ausschlaggebende Teil der Ontologie ist in Abbildung 7.2 dargestellt.

Die Ausprägungen `Good` und `Bad` der Situation `Weather` wurden bereits im Bayes'schen Netz modelliert (vgl. Abschnitt 7.1.2). Daher wurde für diese Situation die Klasse `Weather` mit den beiden Unterklassen `GoodWeather` und `BadWeather` angelegt. Der Domänenexperte definierte anschließend, dass Klassifizierung des Wetters von der Stärke der Sonneneinstrahlung und der Menge des Regens auf der Windschutzscheibe, abhängig ist. Die beiden Klassen `SunIntensity` und `RainIntensity` repräsentieren die dazugehörigen Sensoren. Während die Stärke der Sonneneinstrahlung vom Sensor in Lux angegeben wird, liefert der Sensor für die Regenmenge einen Wert zwischen 0 und 1. Die konkreten Werte werden durch die Data Properties `sunValue` bzw. `rainValue` repräsentiert.

Um die Klassifizierung des Wetters nicht auf Basis der genauen Sensorwerte durchführen zu müssen, wurde vom Domänenexperten festgelegt, dass die Klasse `SunIntensity` die Unterklassen `LowSun`, `MediumSun` und `HighSun` erhält und der Klasse `RainIntensity`, analog dazu, die Unterklassen `NoRain`, `LowRain`, `MediumRain` und `HighRain` zugeordnet werden. Ferner wurde modelliert, dass die Klassifizierung in diese Unterklassen mit Hilfe von Datenbereichen der Data Properties, erfolgt. So

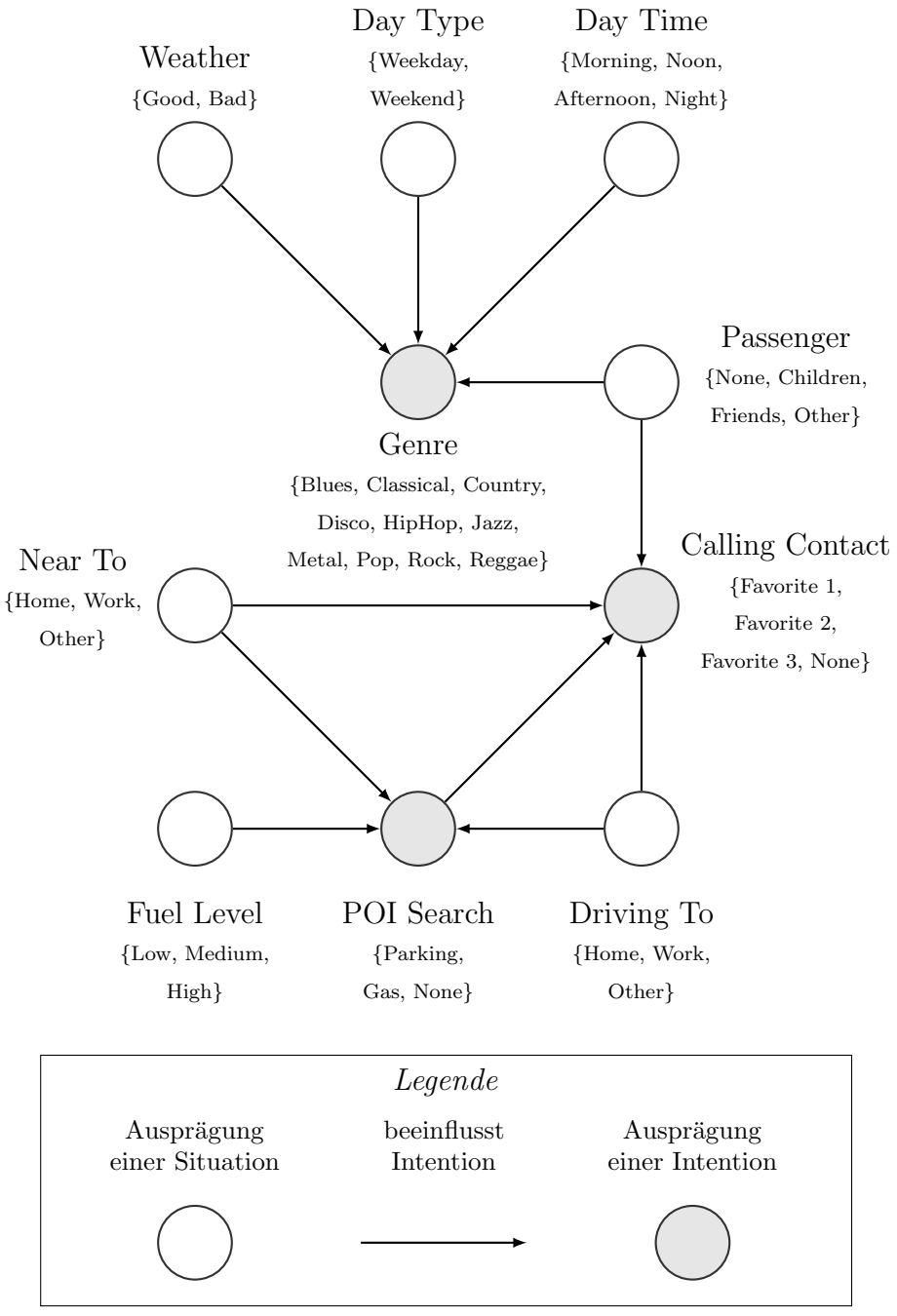

Abbildung 7.1: Struktur des Bayes'schen Netzes für die Fallstudie INIS

Ausprägung `Near To`	Ausprägung `Fuel Level`	Ausprägung `Driving To`	`Parking`	`Gas`	`None`
Home	Low	Home	1/3	1/3	1/3
Home	Low	Work	1/3	1/3	1/3
Home	Low	Other	1/3	1/3	1/3
Home	Medium	Home	1/3	1/3	1/3
Home	Medium	Work	2/10	7/10	1/10
Home	Medium	Other	5/10	4/10	1/10
Home	High	Home	2/10	3/10	5/10
Home	High	Work	1/10	4/10	5/10
Home	High	Other	2/10	7/10	1/10
Work	Low	Home	1/3	1/3	1/3
Work	Low	Work	1/3	1/3	1/3
Work	Low	Other	1/3	1/3	1/3
Work	Medium	Home	1/3	1/3	1/3
Work	Medium	Work	2/10	7/10	1/10
Work	Medium	Other	5/10	4/10	1/10
Work	High	Home	2/10	3/10	5/10
Work	High	Work	1/10	4/10	5/10
Work	High	Other	2/10	7/10	1/10
Other	Low	Home	1/3	1/3	1/3
Other	Low	Work	1/3	1/3	1/3
Other	Low	Other	1/3	1/3	1/3
Other	Medium	Home	1/3	1/3	1/3
Other	Medium	Work	2/10	7/10	1/10
Other	Medium	Other	5/10	4/10	1/10
Other	High	Home	2/10	3/10	5/10
Other	High	Work	1/10	4/10	5/10
Other	High	Other	2/10	7/10	1/10

Tabelle 7.1: Wahrscheinlichkeitsverteilung für die von den Situationen `Near To`, `Fuel Level` und `Driving To` abhängende Nutzerintention `POI Search` (ergänzend zu Abbildung 7.1)

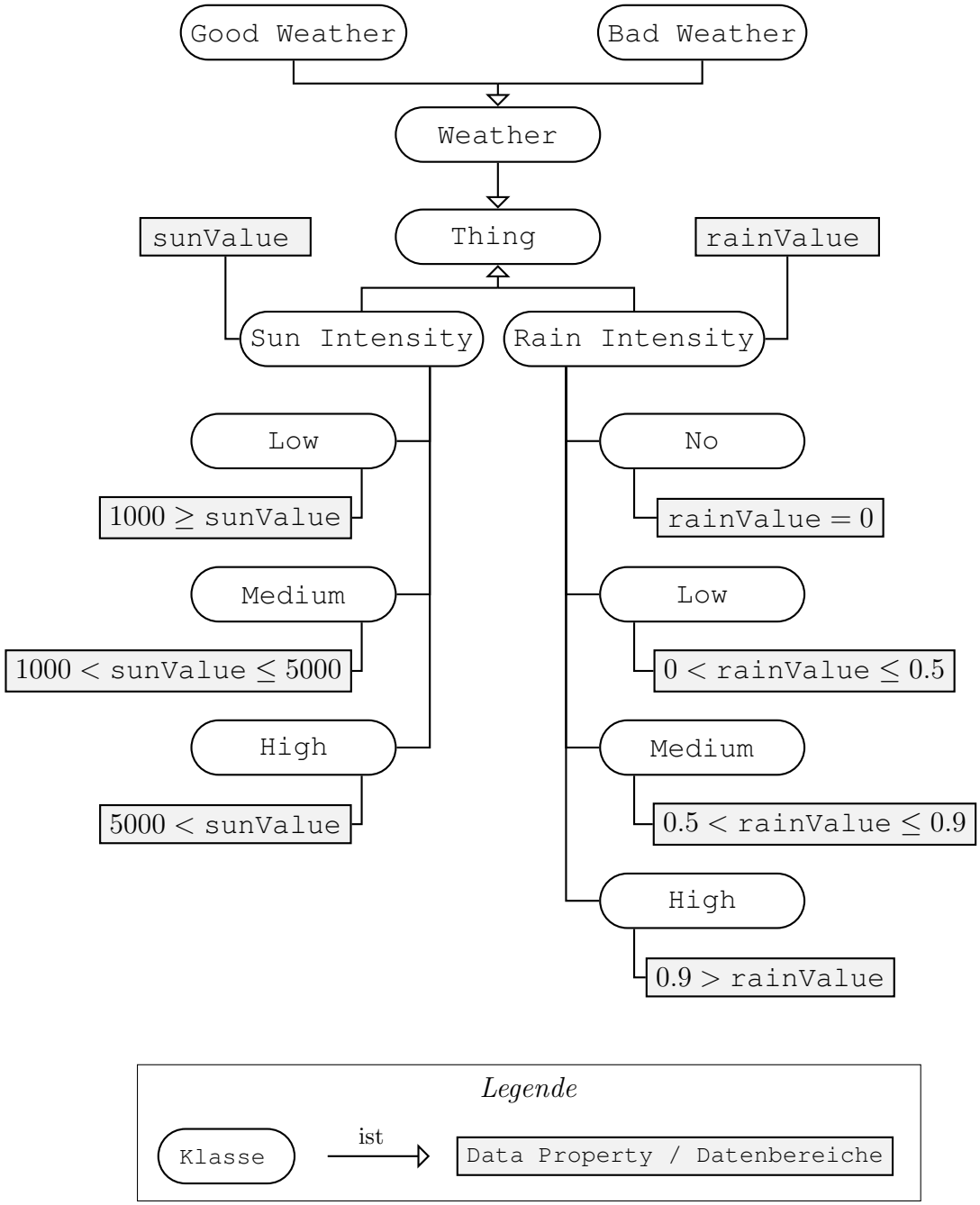

Abbildung 7.2: Auszug zur Situation `Weather` aus der für INIS erstellten Ontologie

wird zur Laufzeit ein Individuum der Klassen `RainIntensity` beispielsweise genau dann der Unterklasse `MediumRain` zugeordnet, wenn der vom Sensor übermittelte Wert (`rainValue`) größer als 0.5 und kleiner oder gleich 0.9 ist.

Für die Klassifizierung des Wetters wurde die Ontologie abschließend vom Modellierer um SWRL-Regeln ergänzt. Zwei dieser Regeln sind exemplarisch in SWRL-Regel 7.1 und SWRL-Regel 7.2 dargestellt.

SWRL-Regel 7.1: *Klassifikation von schlechtem Wetter anhand von Regenmenge und Sonnenstärke*

$\texttt{r} \in \texttt{MediumRain} \land \texttt{s} \in \texttt{LowSun} \land \texttt{w} \in \texttt{Weather} \Rightarrow \texttt{w} \in \texttt{BadWeather}$

SWRL-Regel 7.2: *Klassifikation von gutem Wetter anhand von Regenmenge und Sonnenstärke*

$\texttt{r} \in \texttt{NoRain} \land \texttt{s} \in \texttt{HighSun} \land \texttt{w} \in \texttt{Weather} \Rightarrow \texttt{w} \in \texttt{GoodWeather}$

Während SWRL-Regel 7.1 dazu verwendet wird, ein Individuum der Klasse `Weather` genau dann als `BadWeather` zu klassifizieren, wenn die Regenmenge als mittel (`MediumRain`) und die Sonnenstärke als niedrig (`LowSun`) klassifiziert wurde, so verfolgt SWRL-Regel 7.2 den Zweck, das Individuum als `GoodWeather` zu klassifizieren, wenn kein Regen erkannt (`NoRain`) und eine hohe Sonnenstärke (`HighSun`) gemessen wurde. Auch für alle anderen Kombinationen von Regenmenge und Sonnenstärke wurde modelliert, welche Konsequenzen diese jeweils für das Individuum der Klasse `Weather` bedeutet. Da für die Regenmenge vier und für die Sonnenstärke drei Unterklassen modelliert wurden, ergeben sich dadurch 12 mögliche Kombinationen, die durch entsprechende SWRL-Regeln abgebildet werden müssen. Dies bedeutet jedoch nicht, dass 12 SWRL-Regeln erstellt werden müssen. SWRL-Regel 7.3 zeigt beispielsweise, dass die Kombinationsmöglichkeiten, in denen die Regenmenge als `HighRain` klassifiziert wurde, das Individuum der Klasse `Weather`, ohne Berücksichtigung der Sonnenstärke, als `BadWeather` klassifiziert werden kann. Diese Regel erfasst damit drei der 12 Kombinationsmöglichkeiten.

SWRL-Regel 7.3: *Klassifikation von schlechtem Wetter anhand der Regenmenge*

$\texttt{r} \in \texttt{HighRain} \land \texttt{w} \in \texttt{Weather} \Rightarrow \texttt{w} \in \texttt{BadWeather}$

Die so modellierten Kontextinformationen, Situationen und Nutzerintentionen konnten im Anschluss durch ein wie in Kapitel 6 gezeigtes System dazu verwendet werden, ein FIS erlebbar zu machen, dass

a) auf Basis von Sensorwerten, wie beispielsweise Regenmenge und Sonnenstärke, in der Lage ist, die aktuelle Situation in der sich sein Nutzer und das Fahrzeug befinden zu erkennen und

b) im Stande ist, Muster in den Bedienhandlungen seines Nutzers in Abhängigkeit zur aktuellen Situation zu erkennen und sogar vorherzusagen.

So hatte die gemessene Sonnenstärke und Regenmenge beispielsweise einen Einfluss auf das Genre der gespielten Musik. Änderte sich die Situation (zum Beispiel indem das Wetter als `BadWeather` klassifiziert wurde), wurde dem Nutzer vorgeschlagen ein anderes Musikgenre zu hören. Dieser Vorschlag basierte sowohl auf der durch die Bayes'schen Netze modellierten Wahrscheinlichkeitsverteilungen, als auch den Bedienhandlungen des Nutzers in der Vergangenheit. Es sei an dieser Stelle erneut darauf hingewiesen, dass eine Bewertung der modellierten Abhängigkeiten nicht Bestandteil dieser Arbeit ist. Ob diese Abhängigkeiten tatsächlich der Realität entsprechen, muss durch weitere Arbeiten belegt werden.

Die in diesem Abschnitt vorgestellte Fallstudie dient als Grundlage zum Verständnis des folgenden Abschnittes, der zur Beantwortung der ersten Forschungsfrage untersucht, in welchem Maße die von Domänenexperten aufgestellten Anforderungen durch Ontologien und Bayes'sche Netze erfüllt werden.

7.2 Evaluation der Anforderungen

Zur abschließenden Bewertung der gewählten Techniken zur Modellierung von Kontextinformationen und Nutzerintentionen, wird in diesem Abschnitt analysiert, ob sich die in Kapitel 3 ermittelten Anforderungen unter Berücksichtigung der vorgestellten Fallstudie INIS erfüllen lassen. Tabelle 7.2 fasst die Analyse der in Kapitel 3 ermittelten Anforderungen vorab zusammen und zeigt, welche Anforderungen vollständig, teilweise oder nicht erfüllt werden.

7.2.1 Verarbeitung von Kontextinformationen

Zunächst werden die in Abschnitt 3.3.1 ermittelten Anforderungen hinsichtlich ihrer Erfüllung untersucht.

Anforderung A01: Verknüpfung mit bereits vorhanden Daten

Der in Abschnitt 5.4 vorgestellte Prozess zur Modellierung von Nutzerintentionen und Kontextinformationen erlaubt es, ausgehend von einer Datenbeschreibung das Modell der Kontextinformationen mit im Fahrzeug bereits vorhandenen Daten, wie beispielsweise Sensorwerten oder Daten aus Online-Diensten, zu verknüpfen. INIS wurde anhand dieses Prozesses entwickelt. Dabei konnte beobachtet werden, dass aufgrund des vorgestellten Hilfsmittels der Quellcode-Generierung, solche Verknüpfungen vom Modellierer sehr leicht erstellt werden konnten und keinerlei Vorwissen, wie etwa unter welcher Adresse bestimmte Sensorwerte abrufbar sind, erforderlich war.

Diese Anforderung wird daher durch die im Rahmen dieser Arbeit vorgestellten Techniken und Methoden zur Modellierung von FIS als erfüllt betrachtet.

Anforderung	Erfüllt?
Anforderung A01: Verknüpfung mit bereits vorhanden Daten	✓
Anforderung A02: Abstraktion & Aggregation	✓
Anforderung A03: Abbilden von Regeln	✓
Anforderung A04: Unterstützung verschiedener Datentypen	○
Anforderung A05: Unterstützung verschiedener Änderungsfrequenzen	○
Anforderung A06: Klassifikationen von Kontextinformationen	✓
Anforderung A07: Kontextinformationen aus multiplen Quellen	○
Anforderung A08: Kontextinformationen aus unsicheren Quellen	✓
Anforderung A09: Verarbeitung von Kontextinformationen aus unsicheren Quellen	✓
Anforderung A10: Partielle Auswertung	–
Anforderung A11: Unvollständige Daten	✓
Anforderung A12: Modell als Austausch-Medium	✓
Anforderung A13: Erweiterbarkeit	✓
Anforderung A14: Wahrscheinlichkeit der Nutzerintention	✓
Anforderung A15: Mehrere Nutzerintentionen zu einem Zeitpunkt	✓
Anforderung A16: Nutzerintentionen sind situationsabhängig	✓
Anforderung A17: Anpassung an den individuellen Nutzer	✓
Anforderung A18: Vergessen von gelernten Nutzerintentionen	✓
Anforderung A19: Hohe Verständlichkeit des Modells	○
Anforderung A20: Modell ist standardisiert	✓
Anforderung A21: Vorhandensein einer Tool-Unterstützung	✓
Anforderung A22: Aktualität	✓

✓: vollständig erfüllt; ○ : teilweise erfüllt; – : nicht erfüllt

Tabelle 7.2: Analyse der vorgestellten Techniken und Methoden zur Modellierung von Kontextinformationen und Nutzerintentionen anhand der in Kapitel 3 vorgestellten Anforderungen

Anforderung A02: Abstraktion & Aggregation

Die in Kapitel 4 vorgestellten Möglichkeiten erlauben es, auf Basis von einfachen Sensorwerten, wie beispielsweise der gemessenen Regenmenge auf der Windschutzscheibe des Fahrzeugs, auf höherwertige Kontextinformationen zu schließen. Abschnitt 7.1 zeigt, dass diese Möglichkeiten fundamental wichtig für die Umsetzung von INIS sind und verdeutlicht dazu, wie eine solche Abstraktion anhand einzelner, konkreter Beispiele aussehen kann. So kann eine gemessene Regenmenge auf der Windschutzscheibe beispielsweise abstrakt als *wenig* Regen interpretiert und später zur Beschreibung des Wetters verwendet werden.

Diese Anforderung wird daher durch die im Rahmen dieser Arbeit vorgestellten Techniken und Methoden zur Modellierung von FIS als erfüllt betrachtet.

Anforderung A03: Abbilden von Regeln

Mit dieser Anforderung wurde gefordert, dass ein Modellierer die Möglichkeit hat, regelbasiert über die Abstraktion und Aggregation höherwertiger Kontextinformationen entscheiden zu können. Auch hier konnte INIS zeigen, dass dieser Anforderung, durch die Verwendung von SWRL-Regeln, entsprochen werden kann. So konnte beispielsweise die Tatsache, dass *kein Regen* und *viel Sonne* als *gutes Wetter* zu verstehen ist, durch SWRL-Regel 7.1 modelliert werden.

Diese Anforderung wird daher durch die im Rahmen dieser Arbeit vorgestellten Techniken und Methoden zur Modellierung von FIS als erfüllt betrachtet.

Anforderung A04: Unterstützung verschiedener Datentypen

Die durch diese Anforderung geforderten Datentypen (*Boolean, Integer, Double, String*, sowie aus diesen Datentypen zusammengesetzte komplexe Datentypen), lassen sich mit dem Einsatz von Ontologien verwenden. Während der Erstellung von INIS zeichnete sich jedoch ab, dass einige, wenngleich auch nicht geforderte Datentypen, hilfreich wären, die von Ontologien bzw. dem verwendeten Reasoner *Pellet* nicht oder nur unzureichend unterstützt werden. Dazu zählt beispielsweise der Datentyp *Time*, der für die Situation `Day Time` (vgl. Abbildung 7.1).

Diese Anforderung wird daher als teilweise erfüllt betrachtet.

Anforderung A05: Unterstützung verschiedener Änderungsfrequenzen

Prinzipiell gibt es weder in der Implementierung, noch in der in den jeweiligen Modellen Beschränkungen bezüglich der Änderungsfrequenzen. Sowohl bei der Auswertung Bayes'scher Netze, als auch bei der Auswertung von Ontologien ist jedoch zu beachten, dass diese Auswertung eine gewisse Zeit in Anspruch nimmt. Wird eines der beiden Modelle ausgewertet, so können diesem Modell keine neue Daten (wie zum Beispiel neue Sensorwerte) hinzugefügt werden. Wie Abschnitt 7.3 noch zeigen wird, variiert dieser Zeitraum je nach Größe des Modells. Für die Implementierung von INIS hatte diese Tatsache zur Folge, dass eine zusätzliche Komponente entwickelt werden musste, die überprüft, ob eines der beiden Modelle aktuell ausgewertet wird und nur falls dies nicht der Fall ist, den Modellen zur Laufzeit neue Daten hinzufügt.

Diese Anforderung wird daher als teilweise erfüllt betrachtet.

Anforderung A06: Klassifikationen von Kontextinformationen

INIS zeigt auch auf, dass Ontologien genutzt werden können, um einfache Sensorwerte zu klassifizieren. So kann beispielsweise die gemessene Regenmenge auf der Windschutzscheibe des Fahrzeugs in die Klassen *wenig*, *mittel*, *viel* klassifiziert werden, wodurch die Weiterverarbeitung vereinfacht und vereinheitlicht wird.

Diese Anforderung wird daher als vollständig erfüllt betrachtet.

Anforderung A07: Kontextinformationen aus multiplen Quellen

Eine, durch eine Klasse in einer Ontologie repräsentierte, Kontextinformation kann aus verschiedenen Quellen stammen. Möglich wird dies durch das Hinzufügen weiterer *SourceName Annotationen* (vgl. Abschnitt 4.2). Von der Quellcode-Generierung und der Generierung der Ontologie-Vorlage wird diese Tatsache aktuell jedoch nicht unterstützt. Würde beispielsweise ein zweiter Sensor zur Ermittlung der Regenmenge verbaut sein, so würde für diesen eine zweite Klasse in der Ontologie erzeugt werden.

Da diese Anforderung konzeptionell erfüllbar ist, jedoch nicht umgesetzt wurde, wird diese Anforderung lediglich als teilweise erfüllt betrachtet.

Anforderung A08: Kontextinformationen aus unsicheren Quellen

Die Behandlung von Unsicherheiten erfolgt innerhalb von INIS wie in Abschnitt 4.2.3 beschrieben. Die Unsicherheiten der einzelnen Sensorwerte werden aktuell jedoch statisch während der Modellierung vom Modellierer festgelegt, da die verbauten Sensoren zur Laufzeit keine Auskunft über die Sicherheit ihrer ermittelten Werte geben können.

Diese Anforderung kann jedoch dennoch als erfüllt betrachtet werden, da sowohl die Modelle, als auch die Implementierungen darauf vorbereitet sind, Wahrheitswerte der Sensoren auszuwerten.

Anforderung A09: Verarbeitung von Kontextinformationen aus unsicheren Quellen

Auch die Verarbeitung der zuvor diskutierten Unsicherheiten während der Abstraktion und Aggregation von einfachen Kontextinformationen zu höherwertigen Kontextinformationen wurde in INIS erfolgreich umgesetzt. Wie in Abschnitt 4.2.5 gezeigt, werden dabei Annotationen vom Modellierer verwendet um die Berechnung der Sicherheit einer höherwertigen Kontextinformation (Situation) zur Laufzeit zu berechnen.

Diese Anforderung wird daher ebenfalls als erfüllt betrachtet.

Anforderung A10: Partielle Auswertung

Ontologien erlauben es derzeit nicht, partiell ausgewertet zu werden. Es ist jedoch möglich, nicht nur eine Ontologie zu erstellen, sondern die zu modellierenden Fakten auf mehrere Ontologien zu verteilen. Dadurch wird eine partielle Auswertung leicht möglich. Die in Kapitel 6 vorgestellten Implementierungen sind darauf jedoch aktuell nicht ausgelegt, weshalb eine solche Aufteilung während der Entwicklung von INIS auch nicht angewendet werden konnte.

Diese Anforderung wird daher als nicht erfüllt betrachtet.

Anforderung A11: Unvollständige Daten

Nicht vorhandene Daten führen bei der Auswertung einer Ontologie lediglich dazu, dass bestimmte Klassen nicht klassifiziert werden können. Jedoch auch nur dann, wenn diese Daten für die Klassifizierung notwendig sind. Auf andere Klassifizierungen oder SWRL-Regeln hat das Fehlen dieser Daten keinen Einfluss. Diese Anforderung erwies sich während der Entwicklung von INIS als sehr hilfreich, da dadurch erste Teilumfänge entwickelt und getestet werden konnte, obwohl noch nicht alle benötigten Daten vorlagen. Auch zur Laufzeit des vollständigen Systems konnte beobachtet werden, dass der Wegfall einzelner Informationen keine Auswirkungen auf solche Teile des Systems hatte, die nicht von diesen Informationen abhängig waren. So liefert der Regensensor im Serienzustand beispielsweise nur dann Werte, wenn die Scheibenwischerautomatik vom Nutzer aktiviert wurde. Situationen und Nutzerintentionen konnte jedoch auch erkannt werden, wenn diese nicht eingeschaltet war und somit kein Wert vom Regensensor vorlag.

Diese Anforderung kann daher als vollständig erfüllt betrachtet werden.

Anforderung A12: Modell als Austausch-Medium

Während der Erstellung von INIS zeigte sich, dass die Ontologie als Modell der Kontextinformationen gut geeignet ist, um einen einheitlichen Wissensstand zwischen verschiedenen beteiligten Fachbereichen zur Verfügung zu stellen. So diente die Ontologie als einheitlicher Wissensstand zwischen Bedienkonzept-Entwicklern, System-Entwickler, System-Integrator und Projektleiter.

Diese Anforderung wird daher als erfüllt betrachtet.

Anforderung A13: Erweiterbarkeit

Durch die Verwendung von Quellcode-Generierung war es während der Entwicklung von INIS leicht möglich, auf sich ändernde Anforderung zu reagieren und beispielsweise das Modell der Kontextinformationen anzupassen. Im Anschluss musste im Fahrzeug lediglich das Modell ausgetauscht werden und die neuen Eigenschaften des Systems konnten getestet und erlebt werden. Die Anpassung der Implementierung war zu keiner Zeit notwendig.

Die Anforderung wird daher als vollständig erfüllt betrachtet.

7.2.2 Erkennung von Nutzerintentionen

Im Folgenden werden nun die an die Erkennung von Nutzerintentionen gestellten Anforderungen im Hinblick auf ihrer Erfüllung analysiert.

Anforderung A14: Wahrscheinlichkeit der Nutzerintention

Die wohl zentralste Anforderung forderte, dass das System zur Erkennung von Nutzerintentionen nicht nur in der Lage ist, diese zu erkennen, sondern auch eine Wahrscheinlichkeit für diese mitzuliefern. Die Modellierung durch Bayes'sche Netze und die in Kapitel 6 vorgestellten Implementierungen erlauben es, zu jeder

Zeit eine Wahrscheinlichkeit für alle Nutzerintentionen und all ihren Ausprägungen zu erhalten. INIS war daher in der Lage für alle modellierten Nutzerintentionen eine Wahrscheinlichkeitsverteilung zu berechnen und in einer Entwickler-Ansicht anzuzeigen.

Diese Anforderung wird daher als vollständig erfüllt betrachtet.

Anforderung A15: Mehrere Nutzerintentionen zu einem Zeitpunkt

Da zu jeder Zeit Wahrscheinlichkeitsverteilungen für alle Nutzerintentionen geliefert werden, können auch mehrere Nutzerintentionen zur gleichen Zeit vom System ermittelt werden. INIS stellte diese Tatsache unter Beweis, da in einigen Szenarien beispielsweise gleichzeitig ein Musik-Genre und die Navigation zu einem POI vorgeschlagen wurde.

Diese Anforderung wird daher als vollständig erfüllt betrachtet.

Anforderung A16: Nutzerintentionen sind situationsabhängig

INIS zeigte, dass die Basis der Nutzerintentionen Situationen sind, die durch Ontologien modelliert von Kontextinformationen abhängen. Dadurch wird gezeigt, dass Nutzerintentionen, wie gefordert, situationsabhängig sind und auch als solche modelliert werden können.

Diese Anforderung wird daher als vollständig erfüllt betrachtet.

Anforderung A17: Individuelle Anpassung an Nutzer

INIS verwendet die in Abschnitt 5.3.1 vorgestellten Techniken zum Adaptieren des Modells der Nutzerintentionen während der Benutzung. Während der Präsentation und der Benutzung von INIS konnte beobachtet werden, dass sich die Wahrscheinlichkeitsverteilungen der einzelnen Nutzerintentionen änderten, wenn bestimmte Vorschläge des Systems angenommen oder abgelehnt wurden.

Diese Anforderung wird daher als vollständig erfüllt betrachtet.

Anforderung A18: Vergessen von gelernten Nutzerintentionen

INIS zeichnet die relevanten Bedienhandlungen des Nutzers auf und nutzt diese, um die Intentionen seines Nutzers in Abhängigkeit zur aktuellen Situation zu lernen. Werden Aufzeichnungen aus diesem Speicher gelöscht, kommt das einem *Vergessen* von gelerntem Nutzerverhalten gleich. INIS löscht Aufzeichnungen genau dann aus diesem Speicher, wenn es bestimmte Aktionen automatisiert durchgeführt wird und diese anschließend vom Nutzer wieder rückgängig gemacht wird.

Diese Anforderung kann daher als erfüllt betrachtet werden.

7.2.3 Modellierung

Nach der Analyse der Systeme zur Verarbeitung von Kontextinformationen und zur Erkennung von Nutzerintentionen, wird abschließend noch die Erfüllung der Anforderungen an die jeweiligen Modelle bewertet.

Anforderung A19: Verständlichkeit

Im Falle der Bayes'schen Netze lässt sich sagen, dass diese von Bedienkonzept-Entwickler sehr gut verstanden wurden und sie es dadurch ermöglichten leicht und schnell Nutzerintentionen zu modellieren. Der Umgang mit Ontologien bedarf hingegen etwas mehr Einführung. Insbesondere auch deswegen, weil die vorhandene Tool-Unterstützung, wie beispielsweise Protégé[1], nicht optimal auf die im Rahmen dieser Arbeit vorgestellten Techniken und Prozesse abgestimmt ist.

Diese Anforderung wird daher als teilweise erfüllt betrachtet.

Anforderung A20: Standardisierung

Sowohl Bayes'sche Netze, als auch Ontologien können als standardisiert betrachtet werden. Es gibt für beide Datenformate zur Persistierung, beide lassen sich mit Hilfe von Software-Tools anlegen und bearbeiten, und für beide gibt es Bibliotheken, die in eigenen Implementierungen verwendet werden können, um diese Modelle beispielsweise zur Laufzeit auszuwerten.

Diese Anforderung wird daher als erfüllt betrachtet.

Anforderung A21: Tool-Unterstützung

Durch die bereits diskutierte Standardisierung existiert auch eine Tool-Unterstützung für Bayes'sche Netze und Ontologien. Im Rahmen von INIS wurde zur Erstellung des Bayes'schen Netzes das bereits erwähnte *Java Bayes*[2] eingesetzt. Zur Erstellung der Ontologie wurde auf das ebenfalls bereits erwähnte Tool Protégé zurückgegriffen.

Diese Anforderung wird daher als erfüllt betrachtet.

Anforderung A22: Aktualität

Da das Konzept Bayes'scher Netze auf das Jahr 1763 zurückgeht [BP63], können diese sicher nicht als aktuell bezeichnet werden. Allerdings finden Bayes'sche Netze im Bereich der *Künstlichen Intelligenz* nach wie vor Anwendung, wie die Arbeiten von Ablaßmeier [Abl09] oder Schroven [Sch11] zeigen. Ontologien können hingegen durchaus als aktuell bezeichnet werden. Erste Spezifikationen[3] der auch in dieser Arbeit verwendeten Sprache *OWL* ließen sich bereits im Jahr 2004 finden. Aber auch aktuell findet durch das W3C eine Weiterentwicklung von Ontologien statt. Der letzte Entwurf[4] zu *OWL 2* ist beispielsweise auf Juli 2016 datiert.

Diese Anforderung wird daher insgesamt als erfüllt betrachtet.

Zusammenfassend lässt sich sagen, dass der überwiegende Anteil an Anforderungen durch die in dieser Arbeit vorgestellten Techniken und Methoden zur Modellierung von intentionssensitiven FIS erfüllt werden kann. Die nicht oder nur teilweise erfüllten Anforderungen lassen sich durch leichte Erweiterung an den vorgestellten Konzepten oder Implementierungen ebenfalls vollständig erfüllen.

[1] Protégé Website: http://protege.stanford.edu. Zuletzt überprüft am 19.10.2016.
[2] Java Bayes Website: http://www.cs.cmu.edu/~javabayes/. Zuletzt überprüft am 19.10.2016.
[3] Erste OWL-Spezifikation: https://www.w3.org/TR/2004/REC-owl-semantics-20040210/. Zuletzt überprüft am 19.10.2016.
[4] Aktuellste OWL2-Spezifikation: https://www.w3.org/TR/2016/WD-owl-time-20160712/. Zuletzt überprüft am 19.10.2016.

7.3 Performance von Ontologien

In der Literatur lassen sich viele Arbeiten finden, die zeigen, dass die Auswertungsgeschwindigkeit von Ontologien unter bestimmten Umständen sehr gering ausfallen kann [BN04, WGZP04, HLTB04, ABR09]. In diesem Abschnitt wird daher untersucht, welchen Effekt bestimmte Parameter einer Ontologie auf die zur Auswertung dieser Ontologie benötigten Zeit haben.

Dazu werden zunächst die beiden folgenden Annahmen formuliert, die verdeutlichen, von welchen Parametern vor der Durchführung der Untersuchungen ein Effekt auf die Ausführungsgeschwindigkeit erwartet wird.

Annahme 1: Die Anzahl der Klassen in einer Ontologie wirkt sich negativ auf die Auswertungsgeschwindigkeit aus. Ontologien mit vielen Klassen benötigen zur Auswertung mehr Zeit, als Ontologien mit weniger Klassen.

Annahme 2: Die Verwendung von SWRL-Regeln wirkt sich negativ auf die Auswertungsgeschwindigkeit aus. Ontologien mit mehr Regeln benötigen zur Auswertung der in ihr modellierten Kontextinformationen mehr Zeit, als Ontologien mit weniger SWRL-Regeln.

Um diese beiden Annahmen überprüfen zu können, wurde im Rahmen dieser Arbeit ein Untersuchungsaufbau entwickelt, welcher in Abschnitt 7.3.1 und Abschnitt 7.3.2 beschrieben wird. Die Ergebnisse der Untersuchungen werden abschließend in Abschnitt 7.3.3 präsentiert.

7.3.1 Aufbau & Durchführung

Grundlage für den Untersuchungsaufbau sind die bereits in Kapitel 6 beschriebenen Software-Komponenten, innerhalb der, ebenfalls in Kapitel 6 beschriebenen, Software-Architektur. Innerhalb der Komponente *Context Management* wurden Funktionen hinzugefügt, die es möglich machen, die Dauer für verschiedene Prozesse messen zu können. Dadurch wird es möglich, die beiden folgenden Zeitabschnitte quantifizieren zu können:

- Zeit zum Schlussfolgern: beschreibt die Zeit, die der Reasoner[5] benötigt, um aus den in der Ontologie befindlichen Kontextinformationen auf modellierte Situation schlussfolgern zu können.
- Zeit für die Anfrage: beschreibt die Zeit, die benötigt wird, um die Ergebnisse der Schlussfolgerung abzufragen und die *Situation Supplier* (vgl. Kapitel 6) über diese zu benachrichtigen.

Um die oben genannten Annahmen untersuchen zu können, mussten Ontologien mit verschieden Eigenschaften erstellt werden. Um bei dieser Erstellung möglichst unabhängig und objektiv zu bleiben, wurde im Rahmen dieser Arbeit ein Generator entwickelt, der es erlaubt, zufällige Ontologien mit verschiedenen Eigenschaften zu erzeugen. Die beiden wesentlichen Parameter dieses Generators sind:

[5] Auch für diese Untersuchungen wurde der bereits erwähnte Pellet-Reasoner [SPG+07] verwendet.

- die Anzahl der Klassen, die eine Ontologie beinhaltet, sowie
- die Anzahl der SWRL-Regeln, die eine Ontologie beinhaltet.

Ferner wurde ein weiterer Quellcode-Generator entwickelt, der es erlaubt, auf Basis einer Ontologie die erforderlichen *Context Data Sources* zu generieren. So wird es möglich, die generierte Ontologie mit zufälligen, aber korrekten Daten zur Laufzeit zu befüllen.

Zur Überprüfung der ersten Annahme wurden im Anschluss mit Hilfe dieses Generators Ontologien mit 10, 50, 100, 150 und 200 Klassen erzeugt. Für jede dieser Größen wurden drei zufällige Ontologien generiert, die anschließend in drei unabhängigen Durchläufen jeweils 1000 mal mit Eingangsdaten befüllt, das Reasoning durchgeführt und die Zeiten für Schlussfolgern und Anfragen, wie oben beschrieben, gemessen.

Zur Überprüfung der zweiten Annahme wurden aus allen generierten Ontologien mit 100 Klassen die SWRL-Regeln entfernt. Anschließend wurden die dabei entstandenen Ontologien nach gleichem Muster in drei unabhängigen Durchläufen jeweils 1000 mal mit Eingangsdaten befüllt, das Reasoning durchgeführt und dessen Dauer gemessen.

7.3.2 Verwendete Hardware & Rahmenbedingungen

Die durchgeführten Messungen wurden auf einem handelsüblichen Notebook durchgeführt, welches mit einem Intel® Core™ i5-4300 Prozessor (Taktrate 1.90GHz) und 8GB Arbeitsspeicher ausgestattet war. Als Betriebssystem wurde Microsoft® Windows® 7 eingesetzt. Weiterhin wurde sichergestellt, dass der Rechner keine unnötigen Applikationen ausführte und nicht mit einem Netzwerk verbunden war. Auf diese Weiße konnte verhindert werden, dass die Messergebnisse durch äußere Einflüsse beeinflusst werden.

Es wurde für die Untersuchungen bewusst keine heute verfügbare automotive Hardware verwendet, da davon ausgegangen werden kann, dass die Leistungsfähigkeit von Recheneinheiten in Fahrzeugen weiter steigen wird. Heutige Desktop- bzw. Notebook-Hardware bietet daher eine gute Möglichkeit zu zeigen, mit welchen Ausführungszeiten beim Einsatz der gewählten Technologien in zukünftigen Fahrzeuggenerationen zu rechnen ist.

7.3.3 Ergebnisse

Abbildung 7.3 zeigt, dass die Zeit, die für das Reasoning benötigt wird, mit zunehmender Anzahl der Klassen einer Ontologie exponentiell steigt. Auffällig ist auch, dass sich die dabei gemessene Standardabweichung mit steigender Anzahl an Klassen ebenfalls dramatisch erhöht. So wurden bei Ontologien mit 200 Klassen beispielsweise Zeiten zwischen 2000ms und 1ms gemessen. In der Konsequenz bedeutet dies, dass bei Ontologien mit mehreren Klassen keine zuverlässigen Aussagen über die Dauer des Reasonings gemacht werden können.

Abbildung 7.4 zeigt, dass auch die Dauer, die zur Auswertung der Ontologie benötigt wird, mit zunehmender Anzahl der Klassen exponentiell steigt. Auch die Standardabweichung steigt dabei ebenfalls drastisch an. Die Durchführungen zeigen Dauern zwischen 20ms und 12765ms.

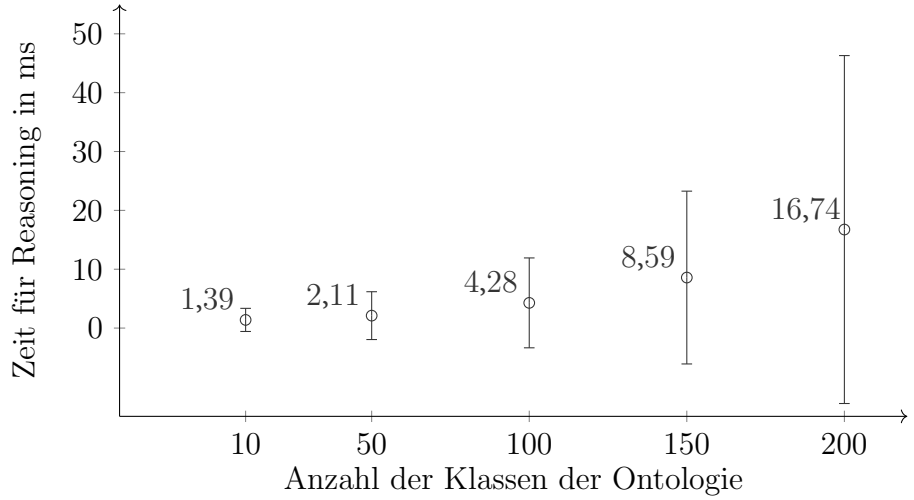

Abbildung 7.3: Durchschnittliche Zeiten und Standardabweichungen die für das Auswerten von Ontologien mit verschiedenen Größen benötigt werden

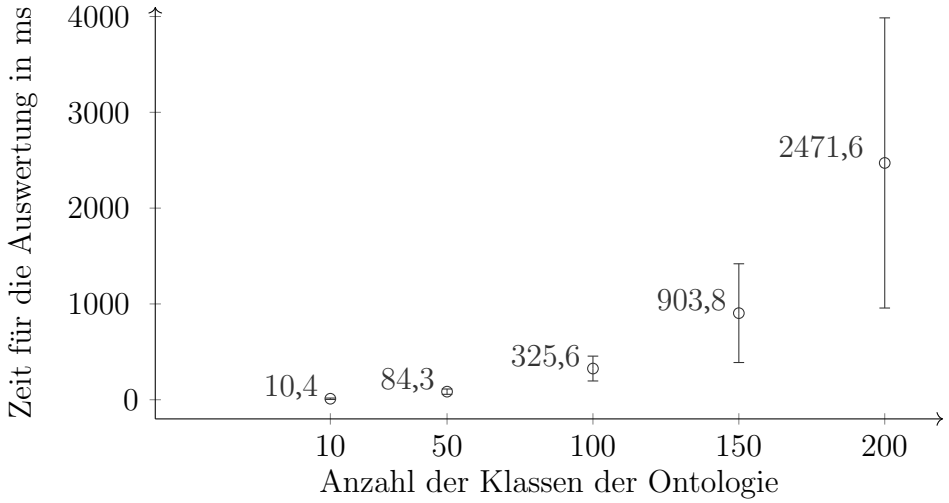

Abbildung 7.4: Durchschnittliche Zeiten und Standardabweichungen die für das Auswerten von Ontologien mit verschiedenen Größen benötigt werden

Abbildung 7.5 zeigt, dass das Reasoning von Ontologien mit SWRL-Regeln genauso schnell durchgeführt werden kann, wie von Ontologien ohne SWRL-Regeln. Gemessen wurden Mittelwerte von 4283ms für Ontologien mit SWRL-Regeln und 4279ms für Ontologien ohne SWRL-Regeln. Die ermittelte Standardabweichung verringerte sich geringfügig, blieb aber auf gleichem Niveau (7636ms bei Ontologien mit SWRL-Regeln und 7044ms bei Ontologien ohne SWRL-Regeln). Die gemessene Höchstdauer bei Ontologien mit SWRL-Regeln betrug 531ms, während bei Ontologien ohne SWRL-Regeln lediglich 369ms gemessen wurden.

Zusammenfassend lässt sich aufgrund dieser Ergebnisse sagen, dass die in Abschnitt 7.3 aufgestellte Annahme 1 bestätigt werden kann. Eine Erhöhung der Anzahl der Klassen innerhalb einer Ontologie führt dazu, dass sich sowohl die Dauer für das Reasoning, als auch die Dauer der Auswertung der Ontologie erhöht werden. Für beide gilt dabei, dass ein exponentielles Wachstum erkennbar ist. Mit Bezug

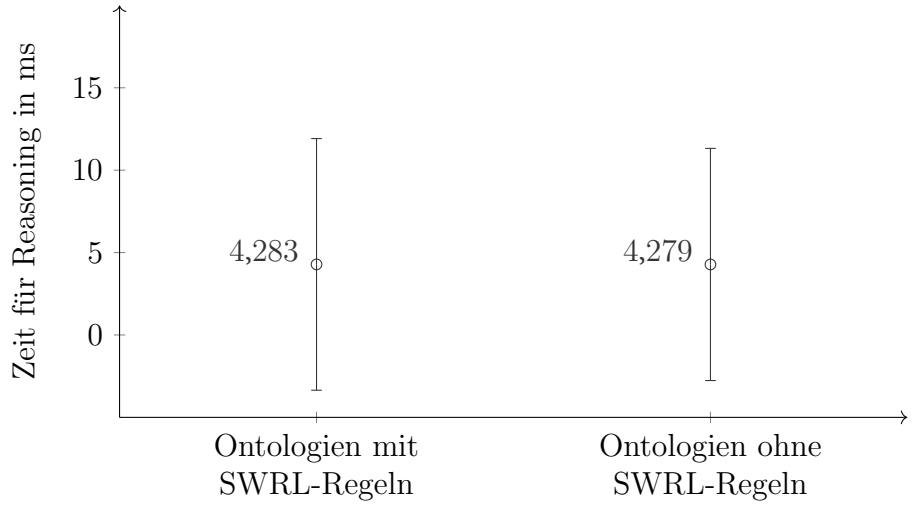

Abbildung 7.5: Durchschnittliche Zeiten und Standardabweichungen, die für das Reasoning von Ontologien mit und ohne SWRL-Regeln benötigt werden

auf die zweite in Abschnitt 7.3 aufgestellte Annahme lässt sich jedoch festhalten, dass der Einsatz von SWRL-Regeln keinerlei negativen Effekt auf die Dauer des Reasoning oder der Auswertung der Ontologie hat. Die zweite Annahme kann daher nicht bestätigt werden.

Entwurfsrichtlinien

Basierend auf den Ergebnissen der vorgestellten Untersuchungen, wurden im Rahmen dieser Arbeit die folgenden Entwurfsrichtlinien aufgestellt. Diese sollen dem Modellierer während seiner Tätigkeit dabei unterstützen, möglichst effiziente Modelle zu erstellen.

1. Prüfung der notwendigen Klassen in einer Ontologie
Aufgrund der Tatsache, dass sowohl das Reasoning, als auch die Auswertung von Ontologien mit mehr Klassen, mehr Zeit in Anspruch nimmt, sollte während der Modellierung darauf geachtet werden, dass unnötige Klassen aus der Ontologie entfernt werden, um diese möglichst schlank zu halten. Eine angepasste Tool-Unterstützung könnte dabei hilfreich unterstützen.

2. Teile und Herrsche
Sollte sich die Anzahl der Klasse unter Berücksichtigung der ersten Richtlinie nicht signifikant reduzieren lassen, so empfiehlt es sich, nicht eine Ontologie mit allen Kontextinformationen und Situationen zu modellieren, sondern die modellierten Informationen in mehrere Ontologien aufzuteilen und diese zur Laufzeit getrennt voneinander auszuwerten. Eine mögliche Aufteilung könnte zum Beispiel drei Ontologien, für die drei Arten von Daten *Fahrer*, *Fahrzeug* und *Umwelt*, sein.

3. SWRL-Regeln nutzen
Abschließend sei ausdrücklich empfohlen, SWRL-Regeln zu verwenden, da durch diese in den durchgeführten Untersuchungen keine Beeinträchtigungen ermittelt werden konnten.

7.4 Validität der Ergebnisse

In diesem Abschnitt wird diskutiert, welche Fehlerquellen für die in Abschnitt 7.2 und Abschnitt 7.3 durchgeführten Untersuchungen existieren und wie die zu erwartenden Fehler minimiert werden können.

Die in Abschnitt 7.2 vorgestellte Evaluation der Anforderungen basierte auf den Erfahrungen und Beobachtungen der in Abschnitt 7.1 vorgestellten Fallstudie INIS. Auch wenn die Anforderungen in Kapitel 3 allgemeingültig formuliert wurden, so besteht die Möglichkeit, dass die Evaluation basierend auf einer Fallstudie keine Schlüsse auf allgemeine kontext- und intentionssensitive FIS zulässt. So könnten andere Fallstudien aufzeigen, dass einzelne Anforderungen weniger stark oder gar nicht erfüllt werden. Um dieser Gefahr vorzubeugen und die angesprochenen Auswirkungen zu minimieren, wurde INIS sehr vielfältig konstruiert. So kommen Kontextinformationen mit unterschiedlichen Eigenschaften zum Einsatz, es wurden die verschiedenen Möglichkeiten zur Modellierung von Abhängigkeiten innerhalb einer Ontologie angewandt und Nutzerintentionen hingen von verschiedenen Kontextinformationen ab.

Die in Abschnitt 7.3 vorgestellte Evaluation zur Performance von Ontologien wurde wie beschrieben mit zufällig generierten Ontologien durchgeführt. Dabei besteht die Gefahr, dass der Generator dieser Ontologien Eigenschaften generiert, die sich auf die zur Ausführung benötigte Zeit auswirkt (positiv oder negativ). Die Entwicklung des Generators ging daher zur Minimierung dieser Gefahr mit einer besonders hohen Qualitätssicherung einher. So wurden beispielsweise zwei Software-Entwickler mit der Aufgabe betreut, unabhängig voneinander verschiedene Teile des Generators zu entwickeln. Die generierten Ontologien wurden anschließend auch einer manuellen Kontrolle unterzogen, um sicherzustellen, dass diese valide sind und die erforderlichen Informationen enthalten. Eine weitere Bedrohung der Validität der Ergebnisse ist in diesem Zusammenhang die verwendete Hardware und Software. Wie bereits erwähnt, wurden die Untersuchungen auf einem handelsüblichen Notebook durchgeführt, auf dem Microsoft Windows 7 lief. Auch wenn andere Applikationen beendet und Netzwerkverbindungen getrennt wurden, kann nicht vollständig ausgeschlossen werden, dass andere aktive Prozesse die Auswertung der Ontologien verlangsamen. Als vorbeugende Maßnahme wurden die Auswertungen der Ontologien daher möglichst oft wiederholt, um etwaige Einflüsse anderer Prozesse raus zu mitteln.

7.5 Zusammenfassung

In diesem Kapitel wurde anhand der Fallstudie INIS gezeigt, wie die in dieser Arbeit vorgestellten Konzepte und Implementierungen zur Modellierung und Ausführung intentionssensitiver FIS verwendet werden können, um eben diese Systeme schnell erlebbar zu machen. Dazu wurden zu Beginn drei Forschungsfragen formuliert, die im weiteren Verlauf des Kapitels beantwortet wurden.

Die erste Forschungsfrage fragte nach der Erfüllung der durch die Domänenexperten aufgestellten und in Kapitel 3 vorgestellten Anforderungen. Die Fallstudie INIS diente zur Beantwortung dieser Frage als Grundlage. In Abschnitt 7.2 wurde gezeigt, dass der überwiegende Teil dieser Anforderungen bei einer Verwendung von Ontologien

und Bayes'schen Netzen zur Modellierung von Kontextinformationen und Nutzerintentionen erfüllt werden kann. Die zweite Forschungsfrage sollte klären, welchen Einfluss eine steigende Anzahl von Kontextinformationen und Nutzerintentionen auf die Auswertungsgeschwindigkeit einer Ontologie hat. Abschnitt 7.3 konnte diese Frage beantworten, indem Untersuchungen präsentiert wurden, die Ontologien mit verschieden Größen systematisch testeten. Dabei wurde deutlich, dass eine steigende Anzahl an Kontextinformationen und Nutzerintentionen negative Auswirkungen auf die zur Auswertung benötigte Zeit haben. Die dritte Forschungsfrage sollte klären, ob sich SWRL-Regeln negativ auf die Auswertungsgeschwindigkeit von Ontologien auswirken. Abschnitt 7.3 beantwortete auch diese Frage. Es konnte dabei gezeigt werden, dass SWRL-Regeln keinen entscheidenden Einfluss auf die zur Auswertung benötigten Zeit einer Ontologie haben.

Unter Berücksichtigung der in diesem Kapitel präsentierten Ergebnisse, wird die Verwendung der vorgeschlagenen Konzepte zur prototypischen Umsetzung kontext- und intentionssensitiver FIS als geeignet eingestuft. Ein Einsatz in anderen Domänen, wie beispielsweise für Fahrerassistenzsysteme, sollte jedoch gesondert betrachtet werden, da diese mitunter zusätzliche Anforderungen an Latenzen und Ausführungsgeschwindigkeiten stellen.

8. Zusammenfassung

„Wenn die anderen glauben man ist am Ende,
so muß man erst richtig anfangen.“

(Konrad Adenauer, erster Deutscher Bundeskanzler)

Der Trend der letzten Jahre zeigt, dass Fahrzeuginformationssysteme einen immer größer werdenden Funktionsumfang beherrschen. Weiterhin wird davon ausgegangen, dass dieser Trend auch die nächsten Jahren anhalten wird. Um ein hohes Maß an Bedienbarkeit zu gewährleisten, wird aktuell untersucht, wie zukünftige Fahrzeuginformationssysteme den Nutzer durch Erkennung seiner Intentionen unterstützen können. Zur Unterstützung dieser Untersuchungen wurden in der vorliegenden Arbeit Modelle, Prozesse und Systeme zur Erkennung des aktuellen Kontext sowie zur Erkennung aktueller Nutzerintentionen vorgestellt und bewertet. Dabei wurde insbesondere auf die Art und Weise der Modellierung solcher Systeme eingegangen.

Zunächst wurden dazu in Kapitel 3 Anforderungen präsentiert, die im Rahmen von Experteninterviews ermittelt wurden. Auf Basis dieser Anforderungen wurden in Kapitel 4 verschiedene Technologien zur Modellierung von Kontextinformationen und Situationen vorgestellt und miteinander verglichen. Dabei stellte sich heraus, dass Ontologien, ergänzt um Objektorientierte-Ansätze, nach aktuellem Stand, eine sehr gute Basis für eine solche Modellierung darstellen. Im weiteren Verlauf desselben Kapitels wurde daher vorgestellt, welche Konzepte im Rahmen der vorliegenden Dissertation erarbeitet wurden, um Kontextinformationen und Situationen zu modellieren.

Anschließend wurden in Kapitel 5 verschiedene Konzepte zur Modellierung von Nutzerintentionen vorgestellt. Als Ergebnis lässt sich dazu festhalten, dass sowohl Künstliche Neuronale Netze, als auch Bayes'sche Netze sehr gute Kandidaten für eine solche Modellierung sind. Aufgrund der höheren Verständlichkeit Bayes'scher Netze wurde im weiteren Verlauf des selben Kapitels gezeigt, wie Nutzerintentionen mit Hilfe Bayes'scher Netze modelliert werden können. Abschließend wurde im selben Kapitel

vorgestellt, wie die beiden Modelle von Kontextinformationen und Nutzerintentionen miteinander verknüpft werden können.

Neben der Modellierung von Kontextinformationen und Nutzerintentionen, zeigt die vorliegende Dissertation mit Kapitel 6 auch, wie die dazugehörige Implementierung eines intentionssensitiven Fahrzeuginformationssystems aussehen kann, das die zuvor erstellen Modelle nutzt, um situationsabhängige Nutzerintentionen zur Laufzeit zu erkennen und diese Erkennung mit der Zeit an die Bedienmuster des Nutzers anpasst.

Die in Kapitel 7 vorgestellte Evaluation verdeutlichte abschließend, dass die vorgestellten Konzepte und Implementierungen zur Realisierung der Fallstudie *INIS* erfolgreich eingesetzt wurden. Sie zeigte auch, dass die Ausführungszeiten von Ontologien für die geplanten Einsatzzwecke hinreichend klein sind, jedoch mit steigender Anzahl modellierter Klassen und je nach Einsatzzweck steigen und damit zu Problemen führen kann. Zusammengefasst kann daher gesagt werden, dass Ontologien für die Modellierung von Kontextinformationen und Situationen zum Einsatz in prototypischen Fahrzeuginformationssystemen geeignet sind. Ebenso eignen sich Bayes'sche Netze, um Nutzerintentionen zu modellieren und zur Laufzeit individuell anzupassen.

Ausblick

Eine mögliche Weiterentwicklung der in dieser Arbeit vorgestellten Konzepte wäre die Kombination von Ontologien mit anderen Techniken zur Modellierung von Kontextinformationen, wie zum Beispiel Künstlichen Neuronalen Netzen. So könnte je nach Anwendungsfall die geeignetste Technik ausgewählt werden. Die vorgestellten Implementierungen bieten dafür bereits eine gute Grundlage, da durch die Entkopplung zwischen dem Modell der Kontextinformationen und dem Modell der Nutzerintentionen entsprechende Schnittstellen vorhanden sind, an denen sich leicht weitere Implementierungen anknüpfen lassen.

Außerdem sollte in weiterführenden Arbeiten untersucht werden, ob sich die nicht oder nur teilweise erfüllten Anforderungen durch Anpassungen an Konzept oder Implementierung nicht auch erfüllen lassen. Insbesondere die Unterstützung verschiedener Datentypen und Änderungsfrequenzen, sowie die partielle Auswertung der Modelle erscheinen für einen potentiellen Serieneinsatz unabdingbar.

A. Schnittstellenbeschreibungen

```
import java.util.ArrayList;
import java.util.List;

public class CMIndividual {

    private String sourcename;
    private String classname;
    private List<CMData> data;

    public CMIndividual(String sourcename, String classname,
                        List<CMData> data) {
        this.sourcename = sourcename;
        this.classname = classname;
        this.data = data;
    }
    public CMIndividual(String sourcename, String classname) {
        this.sourcename = sourcename;
        this.classname = classname;
        data = new ArrayList<CMData>();
    }
    public String getSourcename() {
        return sourcename;
    }
    public String getClassname() {
        return classname;
    }
    public List<CMData> getData() {
        return data;
    }
    public void addData(CMData value) {
        data.add(value);
    }
}
```

Quelltext A.1: Generisches Format zum Austausch von Kontextinformationen als Java-Klasse

```
public class CMData {

    private String dataName;
    private CMSupportedDateType dataType;
    private Object data;

    public CMData(String dataName,
                  CMSupportedDateType dataType,
                  Object data) {
        this.dataName = dataName;
        this.dataType = dataType;
        this.data = data;
    }

    public String getDataName() {
        return dataName;
    }

    public CMSupportedDateType getDataType() {
        return dataType;
    }

    public Object getData() {
        return data;
    }
}
```

Quelltext A.2: Container Werte einer Kontextinformation als Java-Klasse

```
public enum CMSupportedDateType {

    CM_DATE,
    CM_DECIMAL,
    CM_LONG,
    CM_INT,
    CM_TEXT,
    CM_INDIVIDUAL,
    CM_BOOLEAN;

}
```

Quelltext A.3: Unterstütze Datenformate von Werten einer Kontextinformation als Enumeration

```
import org.semanticweb.owlapi.model.OWLClass;
import org.semanticweb.owlapi.util.DefaultPrefixManager;

public abstract interface CMSupplyDataInterface {

    public abstract void onContextChanged(
                                OWLClass outputClass,
                                DefaultPrefixManager prefixManager);

}
```

Quelltext A.4: Schnittstelle als Java-Interface zum Empfang von Klassen einer Ontologie, die eine vorliegende Situation repräsentieren

Liste der Veröffentlichungen

Tagungsbeiträge und Zeitschriftenaufsätze

LÜDDECKE, Daniel ; BERGMANN, Nina ; SCHAEFER, Ina (2014). Ontology-Based Modeling of Context-Aware Systems. *In Proceedings of the 17th International Conference on Model Driven Engineering Languages and System (MODELS)*, Seiten 484–500, Cham, Springer International Publishing.

LÜDDECKE, Daniel ; SEIDL, Christoph ; SCHNEIDER, Jens ; SCHAEFER, Ina (2015). Modeling user intentions for in-car infotainment systems using Bayesian networks. *In Proceedings of ACM/IEEE 18th International Conference on Model Driven Engineering Languages and System (MODELS)*, Seiten 378–385, IEEE.

LÜDDECKE, Daniel ; SEIDL, Christoph ; SCHAEFER, Ina (2015). Efficient Ontology-Based Modeling of Context-Aware In-Car Infotainment Systems – Benchmark Infrastructure and Design Guidelines. *In Proceedings of the International Workshop on Modelling in Automotive Software Engineering (MASE) co-located with ACM/IEEE 18th International Conference on Model Driven Engineering Languages and Systems (MODELS)*, Seiten 23–32, CEUR Workshop Proceedings, CEUR-WS.org.

LÜDDECKE, Daniel ; SEIDL, Christoph ; SCHNEIDER, Jens ; SCHAEFER, Ina (2016). Modeling context-aware and intention-aware in-car infotainment systems. *International Journal on Software and Systems Modeling (SoSyM)*, Seiten 1–15, Berlin, Heidelberg, Springer-Verlag.

Betreute Abschlussarbeiten

BERGMANN, Nina (2014). Modellierung von Kontextinformationen in Fahrzeug-Infotainmentsystemen. *Masterarbeit*, Technische Universität Braunschweig.

REINERT, Stephan (2014). Entwurf und Entwicklung einer Software-Komponente zur Erkennung von Fahrerintentionen in Fahrzeuginformationssystemen. *Bachelorarbeit*, Wolfenbüttel, Ostfalia Hochschule für angewandte Wissenschaften.

PÄTZOLD, Andreas (2015). Optimierung des toolgestützten Modellierungsprozesses kontextsensitiver Infotainmentsysteme. *Bachelorarbeit*, Hochschule Emden/Leer.

Literaturverzeichnis

[Abl09] Ablassmeier, Markus: *Multimodales, kontextadaptives Informationsmanagement im Automobil.* München, Technische Universität München, Dissertation, 2009 (zitiert auf Seite 24 und 110)

[ABR09] Agostini, Alessandra ; Bettini, Claudio ; Riboni, Daniele: Hybrid reasoning in the CARE middleware for context awareness. In: *International Journal of Web Engineering and Technology* 5 (2009), Nr. 1, S. 3. – ISSN 1476–1289 (zitiert auf Seite 18 und 111)

[APRR07] Ablassmeier, Markus ; Poitschke, Tony ; Reifinger, Stefan ; Rigoll, Gerhard: Context-Aware Information Agents for the Automotive Domain Using Bayesian Networks. In: Smith, Michael J. (Hrsg.) ; Salvendy, Gavriel (Hrsg.): *Human Interface and the Management of Information. Methods, Techniques and Tools in Information Design* Bd. 4557. Berlin, Heidelberg : Springer Berlin Heidelberg, 2007. – ISBN 978–3–540–73344–7, S. 561–570 (zitiert auf Seite 24)

[AS97] Akman, Varol ; Surav, Mehmet: The Use of Situation Theory in Context Modeling. In: *Computational Intelligence* 13 (1997), Nr. 3, S. 427–438. – ISSN 1467–8640 (zitiert auf Seite 12)

[AW97] Apté, Chidanand ; Weiss, Sholom: Data MiningData mining with decision trees and decision rules. In: *Future Generation Computer Systems* 13 (1997), Nr. 2, S. 197–210. – ISSN 0167–739X (zitiert auf Seite 65)

[Bar05] Bardram, Jakob E.: The Java Context Awareness Framework (JCAF) - A Service Infrastructure and Programming Framework for Context-Aware Applications. In: *Pervasive '05: Proceedings of the 3rd International Conference on Pervasive Computing.* Springer, 2005. – ISBN 3540260080, S. 98–115 (zitiert auf Seite 15)

[Bau03] Bauer, Joseph: *Identification and modeling of contexts for different information scenarios in air traffic.* Berlin, Technische Universität Berlin, Diplomarbeit, 2003 (zitiert auf Seite 14)

[BBH+10] Bettini, Claudio ; Brdiczka, Oliver ; Henricksen, Karen ; Indulska, Jadwiga ; Nicklas, Daniela ; Ranganathan, Anand ; Riboni, Daniele: A survey of context modelling and reasoning techniques. In: *Pervasive and Mobile Computing* 6 (2010), Nr. 2, S. 161–180. – ISSN 15741192 (zitiert auf Seite xv, 18, 38, 40, 41, 42, 43 und 45)

[BBX97] Brown, P. J. ; Bovey, J. D. ; Xian Chen: Context-aware applications: from the laboratory to the marketplace. In: *IEEE Personal Communications* 4 (1997), Nr. 5, S. 58–64. – ISSN 1070–9916 (zitiert auf Seite 13)

[BC97] Bouzy, Bruno ; Cazenave, Tristan: Using the object oriented paradigm to model context in computer go. In: *Proceedings of the first international and interdisciplinary conference on modeling and using context*, 1997, S. 279–289 (zitiert auf Seite 15)

[Ber14] Bergmann, Nina: *Modellierung von Kontextinformationen in Fahrzeug-Infotainmentsystemen.* Braunschweig, Technische Universität Braunschweig, Masterarbeit, 2014 (zitiert auf Seite xv, 16, 17, 38, 43, 44 und 45)

[BN04] Becker, Christian ; Nicklas, Daniela: Where do spatial context-models end and where do ontologies start? A proposal of a combined approach. In: *UbiComp '04: 6th International Conference on Ubiquitous Computing. Workshop on Advanced Context Modelling, Reasoning and Management* Bd. 3205, Springer, 2004 (Lecture Notes in Computer Science (LNCS)) (zitiert auf Seite 18 und 111)

[Bou95] Bouckaert, Remco R.: *Bayesian Belief Networks: from Construction to Inference.* Utrecht, Netherlands, University of Utrecht, Dissertation, 1995 (zitiert auf Seite 23, 70 und 72)

[BP63] Bayes, Thomas ; Price, Richard: An Essay towards Solving a Problem in the Doctrine of Chances. By the Late Rev. Mr. Bayes, F. R. S. Communicated by Mr. Price, in a Letter to John Canton, A. M. F. R. S. In: *Philosophical Transactions of the Royal Society of London* 53 (1763), Nr. 0, S. 370–418. – ISSN 0261–0523 (zitiert auf Seite 21, 69, 75 und 110)

[BP81] Barwise, Jon ; Perry, John: Situations and Attitudes. In: *The Journal of Philosophy* 78 (1981), Nr. 11, S. 668. – ISSN 0022362X (zitiert auf Seite 12)

[Bre93] Breiman, Leo: *Classification and regression trees.* New York : Chapman & Hall, 1993. – ISBN 9780412048418 (zitiert auf Seite 18)

[But02] Butler, Mark H.: CC/PP and UAProf: Issues, improvements and future directions. In: *Proceedings of W3C Delivery Context Workshop (DIWS 2002)*, 2002 (zitiert auf Seite 14)

[Car68] Carnap, Rudolf: *Einführung in die symbolische Logik: Mit besonderer Berücksichtigung ihrer Anwendungen.* Dritte, unveränderte Auflage. Vienna : Springer Vienna, 1968. – ISBN 3709131413 (zitiert auf Seite 12)

[CEM01] Capra, Licia ; Emmerich, Wolfgang ; Mascolo, Cecilia: Reflective Middleware Solutions for Context-Aware Applications. In: Yonezawa, Akinori (Hrsg.) ; Matsuoka, Satoshi (Hrsg.): *Metalevel Architectures and Separation of Crosscutting Concerns* Bd. 2192. Springer Berlin Heidelberg, 2001. – ISBN 978–3–540–42618–9, S. 126–133 (zitiert auf Seite xvii, 13 und 14)

[CFJ03] Chen, Harry ; Finin, Tim ; Joshi, Anupam: An Ontology for Context-aware Pervasive Computing Environments. In: *The Knowledge Engineering Review* 18 (2003), Nr. 03, S. 197–207. – ISSN 1469–8005 (zitiert auf Seite 17)

[CH92] Cooper, GregoryF. ; Herskovits, Edward: A Bayesian method for the induction of probabilistic networks from data. In: *Machine Learning* 9 (1992), Nr. 4, S. 309–347. – ISSN 0885–6125 (zitiert auf Seite 23, 70 und 72)

[CJ80] Churchill, Winston ; James, Robert R.: *Churchill speaks: Winston S. Churchill in peace and war : collected speeches, 1897-1963.* New York : Chelsea House, 1980. – ISBN 9780877542568 (zitiert auf Seite 25)

[CK00] Chen, Guanling ; Kotz, David: A Survey of Context-Aware Mobile Computing Research / Dartmouth College, Computer Science. Hanover, NH, 2000 (TR2000-381). – Forschungsbericht (zitiert auf Seite 12)

[CMD99] Cheverst, Keith ; Mitchell, Keith ; Davies, Nigel: Design of an object model for a context sensitive tourist GUIDE. In: *Computers & Graphics* 23 (1999), Nr. 6, S. 883–891. – ISSN 00978493 (zitiert auf Seite 15)

[CPFJ04] Chen, Harry ; Perich, Filip ; Finin, Tim ; Joshi, Anupam: SOUPA: Standard Ontology for Ubiquitous and Pervasive Applications. In: *MOBIQUITOUS '04: 1st Annual International Conference on Mobile and Ubiquitous Systems: Networking and Services*, IEEE, 2004. – ISBN 0769522084, S. 258–267 (zitiert auf Seite 17)

[DA99] Dey, Anind K. ; Abowd, Gregory D.: Towards a Better Understanding of Context and Context-Awareness. In: Gellersen, Hans-W (Hrsg.): *Handheld and Ubiquitous Computing* Bd. 1707. Berlin, Heidelberg, Germany : Springer, 1999. – ISBN 978–3–540–66550–2, S. 304–307 (zitiert auf Seite 11)

[DAS01] Dey, Anind ; Abowd, Gregory ; Salber, Daniel: A Conceptual Framework and a Toolkit for Supporting the Rapid Prototyping of Context-Aware Applications. In: *Human-Computer Interaction* 16 (2001), Nr. 2, S. 97–166. – ISSN 0737–0024 (zitiert auf Seite 15)

[DS88] Dutta ; Shekhar: Bond rating: a nonconservative application of neural networks. In: *Neural Networks, 1988., IEEE International Conference on*, 1988, S. 443–450 vol.2 (zitiert auf Seite 21)

[Dwo73] Dworatschek, Sebastian: *Einführung in die Datenverarbeitung: Mit 189 Übungsaufgaben und einem Abbildungsanhang*. 5. Aufl. Berlin [u.a.] : De Gruyter, 1973 (De Gruyter Lehrbuch). – ISBN 9783110042801 (zitiert auf Seite 81)

[EPR11] Eicher, Theo S. ; Papageorgiou, Chris ; Raftery, Adrian E.: Default priors and predictive performance in Bayesian model averaging, with application to growth determinants. In: *Journal of Applied Econometrics* 26 (2011), Nr. 1, S. 30–55. – ISSN 1099–1255 (zitiert auf Seite 24)

[FGG97] Friedman, Nir ; Geiger, Dan ; Goldszmidt, Moises: Bayesian network classifiers. In: *Machine Learning* 29 (1997), Nr. 2-3, S. 131–163. – ISSN 0885–6125 (zitiert auf Seite 23, 70 und 72)

[FPZ05] Fletcher, L. ; Petersson, L. ; Zelinsky, A.: Road scene monotony detection in a fatigue management driver assistance system. In: *IEEE Proceedings. Intelligent Vehicles Symposium*, 2005, S. 484–489 (zitiert auf Seite 7)

[GB10] Glorot, Xavier ; Bengio, Yoshua: Understanding the difficulty of training deep feedforward neural networks. In: *Proceedings of the International Conference on Artificial Intelligence and Statistics (AISTATS'10). Society for Artificial Intelligence and Statistics*, 2010 (zitiert auf Seite 21 und 67)

[GC16] Griol, David ; Callejas, Zoraida: A Neural Network Approach to Intention Modeling for User-Adapted Conversational Agents. In: *Computational intelligence and neuroscience* 2016 (2016). – ISSN 1687–5273 (zitiert auf Seite 21)

[GG01] Ghidini, Chiara ; Giunchiglia, Fausto: Local Models Semantics, or contextual reasoning=locality+compatibility. In: *Artificial Intelligence* 127 (2001), Nr. 2, S. 221–259. – ISSN 00043702 (zitiert auf Seite 13)

[Gho98] Ghosh, Rishab A.: Interviews with Linus Torvalds: What motivates software developers. In: *First Monday* 3 (1998), Nr. 2. – ISSN 13960466 (zitiert auf Seite 79)

[Giu92] Giunchiglia, Fausto: Contextual Reasoning. In: *EPISTEMOLOGIA, SPECIAL ISSUE ON I LINGUAGGI E LE MACCHINE* 345 (1992), S. 345–364 (zitiert auf Seite 13)

[GM93] George, Edward I. ; McCulloch, Robert E.: Variable Selection via Gibbs Sampling. In: *Journal of the American Statistical Association* 88 (1993), Nr. 423, S. 881–889 (zitiert auf Seite 24)

[GN87] Genesereth, Michael R. ; Nilsson, Nils J.: *Logical foundations of artificial intelligence*. Los Altos, Calif : Morgan Kaufmann, 1987. – ISBN 9780934613316 (zitiert auf Seite 16)

[GNS94] Gorr, Wilpen L. ; Nagin, Daniel ; Szczypula, Janusz: Comparative study of artificial neural network and statistical models for predicting student grade point averages. In: *International Journal of Forecasting* 10 (1994), Nr. 1, S. 17–34. – ISSN 01692070 (zitiert auf Seite 21)

[Gru91] Gruber, Thomas R.: *Ontolingua: A Mechanism to Support Portable Ontologies*. Knowledge Systems Laboratory, Stanford University, 1991 (Report (Stanford University. Knowledge Systems Laboratory)) (zitiert auf Seite 17)

[Gru93] Gruber, Thomas R.: A translation approach to portable ontology specifications. In: *Knowledge Acquisition* 5 (1993), Nr. 2, S. 199–220. – ISSN 10428143 (zitiert auf Seite 16 und 17)

[GS01] Gray, Philip ; Salber, Daniel: Modelling and Using Sensed Context Information in the Design of Interactive Applications. In: Goos, Gerhard (Hrsg.) ; Hartmanis, Juris (Hrsg.) ; van Leeuwen, Jan (Hrsg.) ; Little, Murray R. (Hrsg.) ; Nigay, Laurence (Hrsg.): *Engineering for Human-Computer Interaction* Bd. 2254. Berlin, Heidelberg : Springer Berlin Heidelberg, 2001. – ISBN 978–3–540–43044–5, S. 317–335 (zitiert auf Seite 12)

[GWPZ04] Gu, Tao ; Wang, Xiao H. ; Pung, Hung K. ; Zhang, Da Q.: An Ontology-based Context Model in Intelligent Environments. In: *CNDS'04: Proceedings of the Communication Networks and Distributed Systems Modeling and Simulation Conference*, 2004, S. 270–275 (zitiert auf Seite 17 und 18)

[Hal01] Halpin, Terry: *Information Modeling and Relational Databases: From Conceptual Analysis to Logical Design*. Morgan Kaufmann Publishers, 2001 (zitiert auf Seite 14)

[Haw12] Hawking, Stephen W.: *Remember to look up at the stars and not down at your feet*. Cambridge, 2012 (zitiert auf Seite 1)

[HBSS02] Held, Albert ; Buchholz, Sven ; Schill, Alexander ; Schill, Er: Modeling of Context Information for Pervasive Computing Applications. In: *Proceedings of SCI 2002/ISAS 2002*. 2002 (zitiert auf Seite 13)

[HC99] H. Gregersen ; C. S. Jensen: Temporal entity-relationship models-a survey. In: *IEEE Transactions on Knowledge and Data Engineering* 11 (1999), Nr. 3, S. 464–497. – ISSN 1041–4347 (zitiert auf Seite 14)

[HIR02] Henricksen, Karen ; Indulska, Jadwiga ; Rakotonirainy, Andry: Modeling Context Information in Pervasive Computing Systems. In: *Proceedings of the First International Conference on Pervasive Computing*. London, UK, UK : Springer-Verlag, 2002 (Pervasive '02). – ISBN 3–540–44060–7, S. 167–180 (zitiert auf Seite 14)

[HIR03] Henricksen, Karen ; Indulska, Jadwiga ; Rakotonirainy, Andry: Generating Context Management Infrastructure from High-level Context Models. In: *MDM '03: Proceedings of the 4th International Conference on Mobile Data Management* Bd. 2574, Springer, 2003 (Lecture Notes in Computer Science (LNCS)), S. 1–6 (zitiert auf Seite 14)

[HKRS08] Hitzler, Pascal ; Krötzsch, Markus ; Rudolph, Sebastian ; Sure, York: *Semantic Web: Grundlagen*. 1. London : Springer, 2008. – ISBN 3540339949 (zitiert auf Seite 16)

[HLI04] Henricksen, Karen ; Livingstone, Steven ; Indulska, Jadwiga: Towards a hybrid approach to context modelling, reasoning and interoperation. In: *Proceedings of the First International Workshop on Advanced Context Modelling, Reasoning And Management, in conjunction with UbiComp* Bd. 2004, 2004 (zitiert auf Seite 18)

[HLTB04] Horrocks, Ian ; Li, Lei ; Turi, Daniele ; Bechhofer, Sean: The Instance Store: DL Reasoning with Large Numbers of Individuals (DL2004), Whistler, British Columbia, Canada, June 6-8, 2004. In: Haarslev, Volker (Hrsg.) ; Möller, Ralf (Hrsg.): *Proceedings of the 2004 International Workshop on Description Logics (DL2004), Whistler, British Columbia, Canada, June 6-8, 2004* Bd. 104, CEUR-WS.org, 2004 (CEUR Workshop Proceedings) (zitiert auf Seite 111)

[HMOR94] Hill, Tim ; Marquez, Leorey ; O'Connor, Marcus ; Remus, William: Artificial neural network models for forecasting and decision making. In: *International Journal of Forecasting* 10 (1994), Nr. 1, S. 5–15. – ISSN 01692070 (zitiert auf Seite 20 und 21)

[HMRV99] Hoeting, Jennifer A. ; Madigan, David ; Raftery, Adrian E. ; Volinsky, Chris T.: Bayesian Model Averaging: A Tutorial. In: *Statistical Science* 14 (1999), Nr. 4, S. 382–401. – ISSN 08834237 (zitiert auf Seite 24)

[Hoc09] Hoch, Stefan: *Kontextmanagement und Wissensanalyse im kognitiven Automobil der Zukunft*, Technische Universität München, Dissertation, 2009 (zitiert auf Seite 6)

[HPSB+04] Horrocks, Ian ; Patel-Schneider, Peter F. ; Boley, Harold ; Tabet, Said ; Grosof, Benjamin ; Dean, Mike ; W3C (Hrsg.): *SWRL: A Semantic Web Rule Language Combining OWL and RuleML*. http://www.w3.org/Submission/SWRL. Version: 2004 (zitiert auf Seite 17)

[HSM12] Hank, Peter ; Suermann, Thomas ; Müller, Steffen: Automotive Ethernet, a Holistic Approach for a Next Generation In-Vehicle Networking Standard. In: Meyer, Gereon (Hrsg.): *Advanced Microsystems for Automotive Applications 2012*. Berlin, Heidelberg : Springer Berlin Heidelberg, 2012. – ISBN 978–3–642–29672–7, S. 79–89 (zitiert auf Seite 7)

[HSP+03] Hofer, Thomas ; Schwinger, Wieland ; Pichler, Mario ; Leonhartsberger, Gerhard ; Altmann, Josef: Context-Awareness on Mobile Devices - The Hydrogen Approach. In: *HICSS '03: Proceedings of the 36th Hawaii International Conference on System Sciences*, IEEE, 2003. – ISBN 0769518745, S. 292–302 (zitiert auf Seite 15)

[IEE10] IEEE: *Systems and software engineering – Vocabulary.* ISO/IEC/IEEE 24765:2010(E) (zitiert auf Seite 26)

[IRRH03] Indulska, Jadwiga ; Robinson, Ricky ; Rakotonirainy, Andry ; Henricksen, Karen: Experiences in Using CC/PP in Context-Aware Systems. In: Goos, Gerhard (Hrsg.) ; Hartmanis, Juris (Hrsg.) ; van Leeuwen, Jan (Hrsg.) ; Chen, Ming-Syan (Hrsg.) ; Chrysanthis, Panos K. (Hrsg.) ; Sloman, Morris (Hrsg.) ; Zaslavsky, Arkady (Hrsg.): *Mobile Data Management* Bd. 2574. Berlin, Heidelberg : Springer Berlin Heidelberg, 2003. – ISBN 978–3–540–00393–9, S. 247–261 (zitiert auf Seite 13 und 14)

[KH04] Kriens, Peter ; Hargrave, BJ: Listener Pattern Considered Harmful: The "Whiteboard" Pattern. 2004. – Forschungsbericht (zitiert auf Seite 10)

[Kno10] Knoll, Peter: Fahrerassistenzsysteme. In: Reif, Konrad (Hrsg.): *Fahrstabilisierungssysteme und Fahrerassistenzsysteme.* Vieweg+Teubner, 2010. – ISBN 3834897175 (zitiert auf Seite 6)

[Kru11] Kruse, Rudolf: *Computational Intelligence: Eine methodische Einführung in Künstliche Neuronale Netze, Evolutionäre Algorithmen, Fuzzy-Systeme und Bayes-Netze; 43 Tab.* 1. Aufl. Wiesbaden : Vieweg + Teubner, 2011 (Studium). – ISBN 978–3–8348–1275–9 (zitiert auf Seite 21, 22 und 53)

[KTK10] Kannan, Saravanan ; Thangavelu, Arunkumar ; Kalivaradhan, RameshBabu: An Intelligent Driver Assistance System (I-DAS) for Vehicle Safety Modelling using Ontology Approach. In: *International Journal of UbiComp* 1 (2010), Nr. 3, S. 15–29 (zitiert auf Seite 6)

[LBS14] Lüddecke, Daniel ; Bergmann, Nina ; Schaefer, Ina: Ontology-Based Modeling of Context-Aware Systems. In: Dingel, Juergen (Hrsg.) ; Schulte, Wolfram (Hrsg.) ; Ramos, Isidro (Hrsg.) ; Abrahão, Silvia (Hrsg.) ; Insfran, Emilio (Hrsg.): *Model-Driven Engineering Languages and Systems: 17th International Conference, MODELS 2014, Valencia, Spain, September 28 – October 3, 2014. Proceedings.* Cham : Springer International Publishing, 2014. – ISBN 978–3–319–11653–2, S. 484–500 (zitiert auf Seite 37)

[Lea78] Leamer, Edward E.: *Specification searches: Ad hoc inference with nonexperimental data.* Bd. 53. John Wiley & Sons Incorporated, 1978 (zitiert auf Seite 24)

[LSS15] Lüddecke, Daniel ; Seidl, Christoph ; Schaefer, Ina: Efficient Ontology-Based Modeling of Context-Aware In-Car Infotainment Systems: Benchmark Infrastructure and Design Guidelines. In: *Proceedings of the International Workshop on Modelling in Automotive Software Engineering* Bd. 1487. 2015, S. 23–32 (zitiert auf Seite 79 und 96)

[LSSS15] Lüddecke, Daniel ; Seidl, Christoph ; Schneider, Jens ; Schaefer, Ina: Modeling user intentions for in-car infotainment systems using Bayesian networks. In: *Model Driven Engineering Languages and Systems (MODELS), 2015 ACM/IEEE 18th International Conference on*, 2015, S. 378–385 (zitiert auf Seite 59 und 96)

[LSSS16] Lüddecke, Daniel ; Seidl, Christoph ; Schneider, Jens ; Schaefer, Ina: Modeling context-aware and intention-aware in-car infotainment systems. In: *Software & Systems Modeling* (2016), S. 1–15. – ISSN 1619–1374 (zitiert auf Seite 37, 59 und 96)

[Lüd12] Lüddecke, Daniel: *Extraktion von Feature-Modellen aus Implementierungsartefakten*. Magdeburg, Otto-von-Guericke Universität Magdeburg, Masterarbeit, 2012 (zitiert auf Seite 9)

[Maa98] Maass, Henning: Location-aware Mobile Applications Based on Directory Services. In: *Mob. Netw. Appl.* 3 (1998), Nr. 2, S. 157–173. – ISSN 1383–469X (zitiert auf Seite 12)

[MB97] McCarthy, John ; Buvac, Sasa: *Formalizing Context (Expanded Notes)*. 1997 (zitiert auf Seite 12)

[McC93] McCarthy, John: Notes on Formalizing Contexts. In: *Proceedings of the Thirteenth International Joint Conference on Artificial Intelligence*. Chambery, France : Morgan Kaufmann, 1993 (zitiert auf Seite 12)

[MPSP12] Motik, Boris ; Patel-Schneider, Peter F. ; Parsia, Bijan: *OWL 2 Web Ontology Language: Structural Specification and Functional-Style Syntax (Second Edition)*. http://www.w3.org/TR/owl-syntax. Version: 2012 (zitiert auf Seite 18)

[MR94] Madigan, David ; Raftery, Adrian E.: Model Selection and Accounting for Model Uncertainty in Graphical Models Using Occam's Window. In: *Journal of the American Statistical Association* 89 (1994), Nr. 428, S. 1535–1546 (zitiert auf Seite 24)

[MS10] Marscholik, Christoph ; Subke, Peter: *Datenkommunikation im Automobil: Grundlagen, Bussysteme, Protokolle und Anwendungen*. 2., überarb. Aufl. Heidelberg : Hüthig Verlag, 2010 (Hüthig Praxis). – ISBN 3800732750 (zitiert auf Seite 7)

[Nol88] Noland, Kenneth: *Context*. University of Hartford, 1988 (zitiert auf Seite 37)

[ÖA97] Öztürk, Pinar ; Aamodt, Agnar: Towards a Model of Context for Case-based Diagnostic Problem Solving. In: *CONTEXT '97: Proceedings of the 1st International and Interdisciplinary Conference on Modeling and Using Context*, 1997, S. 198–208 (zitiert auf Seite 17)

[Par98] Partsch, H.: *Requirements-Engineering systematisch: Modellbildung für softwaregestützte Systeme*. Berlin : Springer, 1998. – ISBN 9783540643913 (zitiert auf Seite 27)

[Pea85] Pearl, Judea: Bayesian Networks: A Model of Self-Activated Memory for Evidential Reasoning, 1985. In: *Proceedings of the 7th Conference of the Cognitive Science Society, University of California, Irvine, CA*, 1985, S. 329–334 (zitiert auf Seite 21, 69 und 75)

[PR11] Pohl, Klaus ; Rupp, Chris: *Basiswissen Requirements Engineering: Aus- und Weiterbildung zum Certified Professional for Requirements Engineering Foundation Level nach IREB-Standard*. 3. korrigierte Aufl. Heidelberg : dpunkt-Verl., 2011 (ISQL-Reihe). – ISBN 9783898647717 (zitiert auf Seite 27 und 28)

[Qui86] Quinlan, J. R.: Induction of Decision Trees. In: *Machine Learning* 1 (1986), Nr. 1, S. 81–106. – ISSN 0885–6125 (zitiert auf Seite 19)

[Qui93] Quinlan, J. R.: *C4.5: Programs for machine learning*. San Mateo, Calif. : Morgan Kaufmann Publishers, 1993 (The Morgan Kaufmann series in machine learning). – ISBN 9781558602380 (zitiert auf Seite 19)

[Raf88] Raftery, Adrian E.: *Approximate Bayes factors for generalized linear models*. University of Washington, Department of Statistics, 1988 (zitiert auf Seite 24)

[Raf93] Raftery, Adrian E.: Bayesian model selection in structural equation models. In: *Sage Focus Editions* 154 (1993), S. 163 (zitiert auf Seite 24)

[Rau08] Rausch, Mathias: *FlexRay: Grundlagen, Funktionsweise, Anwendung*. München [u.a.] : Hanser, 2008. – ISBN 3446412492 (zitiert auf Seite 7)

[Rei10] Reif, Konrad (Hrsg.): *Sensoren im Kraftfahrzeug*. Wiesbaden : Vieweg+Teubner, 2010. – ISBN 978–3–8348–1315–2 (zitiert auf Seite 5)

[RND10] Russell, Stuart J. ; Norvig, Peter ; Davis, Ernest: *Artificial intelligence: A modern approach*. 3rd ed. Upper Saddle River, NJ : Prentice Hall, 2010 (Prentice Hall series in artificial intelligence). – ISBN 0132071487 (zitiert auf Seite 23)

[Roj96] Rojas, Raúl: *Neural networks: A systematic introduction*. Berlin and New York : Springer-Verlag, 1996. – ISBN 9783540605058 (zitiert auf Seite 20, 21, 59, 65 und 67)

[Rom85] Roman, Gruia-Catalin: A Taxonomy of Current Issues in Requirements Engineering. In: *IEEE Computer* 18 (1985), Nr. 4, S. 14–23 (zitiert auf Seite 27)

[RSP+06] Roussaki, Ioanna ; Strimpakou, Maria ; Pils, Carsten ; Kalatzis, Nikos ; Anagnostou, Miltos: Hybrid context modeling: A location-based scheme using ontologies. In: *PerCom'06: Proceedings of the 4th IEEE Annual Conference on Pervasive Computing and Communications Workshops*, IEEE, 2006. – ISBN 0769525202 (zitiert auf Seite 18)

[Rup01] Rupp, Chris: *Requirements-Engineering und -Management: Professionelle, iterative Anforderungsanalyse für IT-Systeme*. München : Hanser, 2001. – ISBN 9783446216648 (zitiert auf Seite 27 und 28)

[SAW94] Schilit, B. ; Adams, N. ; Want, R.: Context-Aware Computing Applications. In: *Mobile Computing Systems and Applications, 1994. WMCSA 1994. First Workshop on*, 1994, S. 85–90 (zitiert auf Seite 12)

[SB05] Sheng, Quan Z. ; Benatallah, Boualem: ContextUML: A UML-based Modeling Language for Model-driven Development of Context-aware Web Services. In: *ICMB' 05: Proceedings of the International Conference on Mobile Business*, IEEE, 2005. – ISBN 0769523676, S. 206–212 (zitiert auf Seite 14)

[SBG99] Schmidt, Albrecht ; Beigl, Michael ; Gellersen, Hans-W: There is more to context than location. In: *Computers & Graphics* 23 (1999), Nr. 6, S. 893–901. – ISSN 00978493 (zitiert auf Seite 15)

[Sch11] Schroven, Frank: *Probabilistische Situationsanalyse für eine adaptive automatisierte Fahrzeuglängsführung*. Logos-Verlag, 2011. – ISBN 3832530150 (zitiert auf Seite 24 und 110)

[Sch15] Schmidhuber, Jürgen: Deep learning in neural networks: An overview. In: *Neural Networks* 61 (2015), S. 85–117. – ISSN 0893–6080 (zitiert auf Seite 21, 67 und 68)

[Sco09] Scott, Michael L.: *Programming language pragmatics*. Third edition. New York, NY : Elsevier, 2009. – ISBN 0123745144 (zitiert auf Seite 12)

[Sie93] Siegmund, Gerd: *Grundlagen der Vermittlungstechnik*. 2., überarb. und erw. Aufl. Heidelberg : V. Decker, 1993 (Reihe Kommunikation & Technik). – ISBN 9783768548922 (zitiert auf Seite 8)

[SLP04] Strang, Thomas ; Linnhoff-Popien, Claudia: A Context Modeling Survey. In: *Proceedings of the 6th International Conference on Ubiquitous Computing. Workshop on Advanced Context Modelling, Reasoning and Management* Bd. 3205, Springer, 2004 (Lecture Notes in Computer Science (LNCS)) (zitiert auf Seite xv, 12, 13, 14, 15, 17, 38, 39, 40, 41 und 45)

[SM64] Sonquist, John A. ; Morgan, James N.: *The detection of interaction effects; a report on a computer program for the selection of optimal combinations of explanatory variables*. Survey Research Center, Institute for Social Research, University of Michigan, 1964 (Survey Research Center, University of Michigan. Monographno. 35) (zitiert auf Seite 19, 59 und 61)

[Spe88] Specht, Donald F.: Probabilistic neural networks for classification, mapping, or associative memory. In: *Neural Networks, 1988., IEEE International Conference on*, 1988, S. 525–532 vol.1 (zitiert auf Seite 21 und 67)

[Spe90] Specht, Donald F.: Probabilistic neural networks. In: *Neural Networks* 3 (1990), Nr. 1, S. 109–118. – ISSN 0893–6080 (zitiert auf Seite 67)

[SPG+07] Sirin, Evren ; Parsia, Bijan ; Grau, Bernardo C. ; Kalyanpur, Aditya ; Katz, Yarden: Pellet: A practical OWL-DL reasoner. In: *Software Engineering and the Semantic Web* 5 (2007), Nr. 2, S. 51–53. – ISSN 1570–8268 (zitiert auf Seite 84 und 111)

[ST94] Schilit, Bill N. ; Theimer, Marvin M.: Disseminating active map information to mobile hosts. In: *IEEE Network* 8 (1994), Nr. 5, S. 22–32. – ISSN 0890–8044 (zitiert auf Seite 11)

[Str03] Strang, Thomas: *Service Interoperability in Ubiquitous Computing Environments*, Ludwig-Maximilians-Universität München, Dissertation, 2003 (zitiert auf Seite 17)

[STW93] Schilit, Bill N. ; Theimer, Marvin M. ; Welch, Brent B.: Customizing mobile applications. In: *Proceedings of USENIX Mobile & Location-Independent Computing Symposium*. Massachusetts : USENIX Association, 1993, S. 129–138 (zitiert auf Seite 12)

[SV95] Singh, Moninder ; Valtorta, Marco: Construction of Bayesian network structures from data: A brief survey and an efficient algorithm. In: *International Journal of Approximate Reasoning* 12 (1995), Nr. 2, S. 111–131. – ISSN 0888613X (zitiert auf Seite 23, 70 und 72)

[Sv01] Schmidt, A. ; van Laerhoven, K.: How to build smart appliances? In: *IEEE Personal Communications* 8 (2001), Nr. 4, S. 66–71. – ISSN 1070–9916 (zitiert auf Seite 15)

[SZ13] Schäuffele, Jörg ; Zurawka, Thomas: *Automotive Software Engineering*. 5. Springer Vieweg, 2013. – ISBN 3834893684 (zitiert auf Seite 7)

[UG+96] Uschold, Mike ; Gruninger, Michael u. a.: Ontologies: Principles, methods and applications. In: *Knowledge engineering review* 11 (1996), Nr. 2, S. 93–136 (zitiert auf Seite 16)

[Ulr96] Ulrich, Lars: In: *So What!* (1996) (zitiert auf Seite 59)

[VB96] Voelker, GeoffreyM. ; Bershad, BrianN.: Mobisaic: An Information System for a Mobile Wireless Computing Environment. In: Imielinski, Tomasz (Hrsg.) ; Korth, HenryF (Hrsg.): *Mobile Computing* Bd. 353. Springer US, 1996. – ISBN 978–0–7923–9697–0, S. 375–395 (zitiert auf Seite 12)

[vW96] von Doderer, Heimito ; Weber, Dietrich: *Beck'sche Reihe*. Bd. BsR 1158: *Repertorium: Ein Begreifbuch von höheren und niederen Lebens-Sachen*. 2., unveränderte Aufl. München : C.H. Beck, 1996. – ISBN 9783406392580 (zitiert auf Seite 95)

[WGZP04] Wang, Xiao H. ; Gu, Tao ; Zhang, Da Q. ; Pung, Hung K.: Ontology Based Context Modeling and Reasoning using OWL. In: *PerCom'04: Proceedings of the 2nd IEEE Annual Conference on Pervasive Computing and Communications Workshops*, IEEE, 2004. – ISBN 0769521061, S. 18–22 (zitiert auf Seite 17, 18 und 111)

[WHW09] Winner, Hermann ; Hakuli, Stephan ; Wolf, Gabriele: *Handbuch Fahrerassistenzsysteme: Grundlagen, Komponenten und Systeme für aktive Sicherheit und Komfort*. 1. Vieweg+Teubner, 2009. – ISBN 383488619X (zitiert auf Seite 6)

[WRK95] Wang, Richard Y. ; Reddy, M. P. ; Kon, Henry B.: Toward quality data: An attribute-based approach. In: *Decision Support Systems* 13 (1995), Nr. 3-4, S. 349–372. – ISSN 01679236 (zitiert auf Seite 14)

[Wüt08] Wütherich, Gerd: *Die OSGi service platform: Eine Einführung mit Eclipse Equinox*. 1. Aufl. Heidelberg : Dpunkt-Verlag, 2008. – ISBN 3864916275 (zitiert auf Seite xiii, 9 und 10)

[Yeh82] Yeh, R. T.: Requirements analysis—a management perspective. In: *Proceedings of COMPSAC '82*. Chicago, November 1982, S. 410–416 (zitiert auf Seite 27)

[Zim07] Zimmer, Tobias H.: *Verbesserung der Kontexterkennung in Ubiquitären Informationsumgebungen*, Technische Universität Braunschweig, Dissertation, 2007 (zitiert auf Seite 15)

[ZS14] Zimmermann, Werner ; Schmidgall, Ralf: *Bussysteme in der Fahrzeugtechnik: Protokolle, Standards und Softwarearchitektur*. 5., aktualisierte und erw. Aufl. 2014. Wiesbaden : Springer Vieweg, 2014 (SpringerLink : Bücher). – ISBN 3658024194 (zitiert auf Seite 7)

[ZWH04] Zhang, Da Q. ; Wang, Xiao H. ; Hackbarth, Kai: OSGi Based Service Infrastructure for Context Aware Automotive Telematics. In: *VTC '04: 59th IEEE Vehicular Technology Conference* Bd. 5, IEEE, 2004. – ISBN 0780382552, S. 2957–2961 (zitiert auf Seite 7)

Hiermit erkläre ich, dass ich die vorliegende Arbeit selbständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel verwendet habe.

Braunschweig, den 21.10.2016